



APUSIC
固若长城
睿比世界

用戶手冊

金蝶Apusic應用伺服器V10標準版

版權所有 © 深圳市金蝶天燕雲計算股份有限公司2026。保留所有權利。

版權聲明

本文件所涉及的軟件著作權、版權等知識產權已依法進行了註冊，由金蝶天燕雲計算股份有限公司合法擁有。受《中華人民共和國著作權法》《計算機軟件保護條例》《知識產權保護條例》和相關國際版權條約、法律、法規以及其它知識產權法律和條約的保護。未經授權許可，不得非法使用。

免責聲明

本文件包含的版權資訊由金蝶天燕雲計算股份有限公司合法擁有，受法律的保護，金蝶天燕雲計算股份有限公司對本文件可能涉及到的非金蝶天燕雲計算股份有限公司的資訊不承擔任何責任。在法律允許的範圍內，您可以查閱並僅能夠在《中華人民共和國著作權法》規定的合法範圍內複製和打印本文件。任何單位和個人未經金蝶天燕雲計算股份有限公司書面授權許可，不得使用、修改、再發佈本文件的任何部分和內容，否則將被視為侵權，金蝶天燕雲計算股份有限公司有依法追究其責任的權利。

本文件如有更新，不另行通知。對本文件中的問題您可向金蝶天燕雲計算股份有限公司告知或查詢。未經本公司明確授予的任何權利均予保留。

商標聲明

 是深圳市金蝶天燕雲計算股份有限公司向中華人民共和國國家商標局申請註冊的註冊商標，註冊商標專用權由金蝶天燕合法擁有，受法律保護。未經金蝶天燕的書面許可，任何單位及個人不得以任何方式或理由對該商標的任何部分進行使用、複製、修改、傳播、抄錄或與其它產品捆綁使用銷售。凡侵犯金蝶天燕商標權的，金蝶天燕將依法追究其法律責任。本文件提及的其他所有商標或註冊商標，由各自的所有人擁有。

目錄

- 1 前言
 - 1.1 適用對象
 - 1.2 相關文件
 - 1.3 技術支援
- 2 版本變更說明
- 3 產品概述
 - 3.1 產品簡介
 - 3.1.1 概述
 - 3.1.2 平台環境
 - 3.1.3 系統環境
 - 3.1.4 JakartaEE規範
 - 3.1.5 術語
 - 3.2 產品清單
- 4 安裝與執行
 - 4.1 安裝與解除安裝
 - 4.1.1 Java環境
 - 4.1.2 安裝
 - 4.1.3 解除安裝
 - 4.2 許可證授權
 - 4.2.1 普通授權
 - 4.2.2 集中授權
 - 4.2.3 獲取特徵碼
 - 4.2.4 嵌入式部署方式獲取特徵碼
 - 4.3 啟動與停止
 - 4.3.1 在 Linux 環境下啟動
 - 4.3.2 在 Windows 環境下啟動
 - 4.3.3 在 Linux 環境下停止
 - 4.3.4 在 Windows 環境下停止
 - 4.4 訪問管控平台
- 5 應用部署
 - 5.1 自動部署
 - 5.2 指定路徑部署
 - 5.3 手動部署多個應用
 - 5.4 熱部署配置
 - 5.5 配置文根對映
 - 5.6 取消部署
- 6 配置與運維
 - 6.1 配置文件總覽
 - 6.2 設置 JVM 記憶體
 - 6.3 配置數據源
 - 6.3.1 通用配置
 - 6.3.2 開啟耗時SQL語句追蹤
 - 6.3.3 配置連線洩漏回收
 - 6.3.3.1 connectionAutoReclaim配置
 - 6.3.3.2 removeAbandoned配置
 - 6.3.3.3 配置手動回收洩漏連線
 - 6.4 日誌配置
 - 6.4.1 系統日誌配置
 - 6.4.2 配置訪問日誌

- 6.4.3 配置日誌推送
 - 6.4.3.1 類路徑修改
 - 6.4.3.2 配置
- 6.5 端口修改
- 6.6 開啟 JMX
- 6.7 開啟優雅停機
- 6.8 配置慢線程堆疊跟蹤
- 6.9 共享庫配置
- 7 安全配置
 - 7.1 配置 HTTPS
 - 7.1.1 單向認證 -- 證書格式為 JKS
 - 7.1.2 雙向認證 -- 證書格式為 JKS
 - 7.1.3 單向認證 -- 證書格式為 PEM
 - 7.1.4 雙向認證 -- 證書格式為 PEM
 - 7.2 國密配置說明
 - 7.2.1 環境說明
 - 7.2.2 應用伺服器證書配置
 - 7.2.2.1 單向認證
 - 7.2.2.2 雙向認證
 - 7.2.2.3 增加 endpoint 到 server
 - 7.2.3 啟動應用伺服器
 - 7.2.4 客戶端配置
 - 7.2.5 其他問題
 - 7.3 慢攻擊防護
- 8 集中配置
 - 8.1 使用 apollo 集中配置
 - 8.1.1 新增集中配置服務
 - 8.1.2 在 apollo 的配置
 - 8.1.2.1 apusic.conf 配置
 - 8.1.2.2 數據源的配置
 - 8.1.2.3 日誌的配置
 - 8.1.2.4 JVM 的配置
 - 8.2 使用 ETCD 集中配置
 - 8.2.1 新增集中配置服務
 - 8.2.2 在 ETCD 的配置
 - 8.2.2.1 apusic.conf 的配置
 - 8.2.2.2 數據源的配置
 - 8.2.2.3 日誌的配置
 - 8.2.2.4 JVM 的配置
- 9 Session 儲存與叢集
 - 9.1 配置 Session 儲存
 - 9.1.1 配置本地記憶體儲存
 - 9.1.2 配置本地文件儲存
 - 9.1.3 配置資料庫儲存
 - 9.2 叢集方案
 - 9.2.1 Session 組播同步
 - 9.2.1.1 新增叢集配置
 - 9.2.1.2 部署應用和設置 jvmRoute
 - 9.2.1.3 配置 Apache
 - 9.2.1.4 驗證叢集
 - 9.2.2 Session 集中式儲存

- 9.2.2.1 構建 Redis 外掛
 - 9.2.2.1.1 系統構建
 - 9.2.2.2 新增 Redis 配置
- 10 嵌入式部署（與 Spring Boot 整合使用）
 - 10.1 前言
 - 10.1.1 面向對象
 - 10.1.2 概述
 - 10.1.3 產品規範與環境要求
 - 10.2 產品清單
 - 10.2.1 產品清單列表
 - 10.2.2 產品清單說明
 - 10.3 安裝
 - 10.3.1 基於原始碼方式構建
 - 10.3.1.1 匯入單個jar
 - 10.3.1.1.1 前置條件
 - 10.3.1.1.2 安裝資源依賴
 - 10.3.1.1.3 引入資源依賴
 - 10.3.1.1.4 驗證安裝
 - 10.3.1.2 按模組匯入方式
 - 10.3.1.2.1 前置條件
 - 10.3.1.2.2 安裝資源依賴
 - 10.3.1.2.3 引入資源依賴
 - 10.3.1.2.4 驗證安裝
 - 10.3.2 基於可執行jar替換
 - 10.3.2.1 前置條件
 - 10.3.2.2 刪除棄用依賴
 - 10.3.2.3 引入資源依賴
 - 10.3.2.3.1 匯入單個jar
 - 10.3.2.3.2 按模組匯入
 - 10.4 授權認證
 - 10.4.1 獲取特徵碼
 - 10.4.2 專案目錄啟動（未打包狀態）
 - 10.4.3 引數設置啟動（未打包狀態）
 - 10.4.4 專案目錄啟動（可執行jar包狀態）
 - 10.4.5 引數設置啟動（可執行jar包狀態）
 - 10.4.6 集中授權
 - 10.5 容器功能說明
 - 10.5.1 安全管理
 - 10.5.1.1 HTTPS
 - 10.5.1.1.1 簡介
 - 10.5.1.1.2 配置方法
 - 10.5.1.2 國密配置
 - 10.5.1.2.1 簡介
 - 10.5.1.2.2 配置方法
 - 10.5.1.2.3 客戶端配置
 - 10.5.1.2.4 其他問題
 - 10.5.2 國際化支援
 - 10.5.2.1 簡介
 - 10.5.2.2 配置方法
 - 10.5.3 日誌管理
 - 10.5.3.1 應用日誌

- 10.5.3.1.1 簡介
- 10.5.3.1.2 使用方法
- 10.5.3.2 訪問日誌
 - 10.5.3.2.1 簡介
 - 10.5.3.2.2 使用方法
- 10.5.4 監控支援
 - 10.5.4.1 簡介
 - 10.5.4.2 使用方法
 - 10.5.4.2.1 springboot1.x 的版本
 - 10.5.4.2.2 springboot2.x 的版本
- 10.6 示例操作
 - 10.6.1 環境準備
 - 10.6.2 AAS部署
 - 10.6.3 應用工程
 - 10.6.4 pom文件
 - 10.6.5 構建應用包
 - 10.6.6 執行應用
 - 10.6.7 訪問應用
 - 10.6.8 以除錯模式執行
 - 10.6.9 停止應用
- 11 鏈路追蹤與調用鏈分析
 - 11.1 與 Pinpoint 整合
 - 11.1.1 簡介
 - 11.1.2 相關資源
 - 11.1.3 Pinpoint-agent安裝
 - 11.1.4 應用伺服器APM外掛配置
 - 11.1.5 應用伺服器配置
 - 11.1.6 啟動應用伺服器
 - 11.1.7 嵌入式部署配置
 - 11.1.8 Pinpoint APM操作
 - 11.2 與 SkyWalking 整合
 - 11.2.1 簡介
 - 11.2.2 相關資源
 - 11.2.3 Java Agent for Apache SkyWalking安裝
 - 11.2.4 應用伺服器APM外掛配置
 - 11.2.5 應用伺服器配置
 - 11.2.6 啟動應用伺服器
 - 11.2.7 嵌入式部署配置
 - 11.2.8 Skywalking APM操作
 - 11.2.9 問題定位
 - 11.3 調用鏈分析
 - 11.3.1 配置說明
 - 11.3.2 操作方法
- 12 附錄
 - 12.1 附錄A listeners配置說明
 - 12.2 附錄B resources配置說明
 - 12.3 附錄C executors 配置說明
 - 12.4 附錄D endpoints配置說明
 - 12.4.1 SSLHostConfig配置引數
 - 12.4.2 Certificate配置引數
 - 12.5 附錄E servers 配置說明

- 12.5.1 Realm
- 12.5.2 host
- 12.5.3 application
- 12.5.4 servers
- 12.6 附錄F containers 配置說明
- 12.7 附錄G Resource數據源配置說明
- 12.8 附錄H Web通用配置
- 12.9 附錄I 嵌入式部署日誌引數
- 12.10 附錄J relaxed-path-chars和relaxed-query-chars配置示例
 - 12.10.1 Linux
 - 12.10.1.1 環境變數
 - 12.10.1.2 JVM引數
 - 12.10.2 Spring boot配置文件
 - 12.10.2.1 application.properties
 - 12.10.2.2 application.yaml
- 12.11 附錄K actuator詳細說明
 - 12.11.1 配置
 - 12.11.2 端口
 - 12.11.2.1 beans
 - 12.11.2.2 health
 - 12.11.2.3 conditions
 - 12.11.2.4 configprops
 - 12.11.2.5 env
 - 12.11.2.6 loggers
 - 12.11.2.6.1 設置日誌級別
 - 12.11.2.7 threaddump
 - 12.11.2.8 metrics
 - 12.11.2.8.1 所有指標相關說明
 - 12.11.2.9 scheduledtasks
 - 12.11.2.10 mappings

1 前言

本文件是金蝶Apusic應用伺服器V10標準版的用戶手冊，詳細介紹標準版的基本安裝以及配置，提供基本操作方法以及各功能的具體步驟。

1.1 適用對象

本手冊主要面向對象為使用金蝶Apusic應用伺服器進行應用開發的開發人員，生產環境的系統管理員，應用發布人員，技術運維人員等。具備以下技能有助於更好地理解和使用金蝶Apusic應用伺服器用戶手冊內容：

- 熟悉Linux常用命令
- 基本的系統管理任務
- 安裝和管理軟體

1.2 相關文件

瞭解更多AAS-V10標準版產品相關的資訊，請參閱以下AAS-V10產品手冊文件集：

序號	手冊文檔	說明
1	金蝶Apusic應用伺服器V10標準版安裝手冊	詳細介紹如何在各作業系統上安裝AAS，以及AAS服務啟停操作，產品的註冊過程
2	金蝶Apusic應用伺服器V10標準版快速入門	簡單介紹瞭如何快速上手使用AAS
3	金蝶Apusic應用伺服器V10標準版用戶手冊	詳細介紹AAS相關功能的使用和操作說明，指導用戶對系統進行安裝部署、功能操作、參數配置、日常運維及問題自查
4	金蝶Apusic應用伺服器V10標準版管控台用戶手冊	詳細介紹AAS管控台相關功能的使用和操作說明，指導用戶通過可視化管控台進行操作
5	金蝶Apusic應用伺服器V10標準版命令列操作指南	詳細介紹AAS通過命令列進行操作使用
6	金蝶Apusic應用伺服器V10標準版監控指標文件	詳細介紹AAS各類型監控指標
7	金蝶Apusic應用伺服器V10標準版API文件	詳細介紹AAS的API資訊

1.3 技術支援

金蝶Apusic應用伺服器產品提供全面的技術支援服務，您可以通過以下方式獲得技術支援：

- 網址：www.apusic.com
- 電話：400-855-5800
- 郵箱：support@apusic.com
- 金蝶雲社群：<https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

您在取得技術支援時，請提供如下資訊：

1. 您的姓名
2. 公司資訊與聯絡方式
3. 作業系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細資訊

2 版本變更說明

本手冊根據產品實際更新情況同步更新，最新版本將會包括歷史版本內容或作出對應的修改說明。

日期	手冊版本	適用產品	更新說明
2026年5月	V10P11A01F01	AAS標準版 V10	新增慢攻擊配置說明 新增熱部署說明 新增資料庫連線池洩漏回收說明 新增優雅停機說明 更新系統日誌配置說明
2025年12月	V10P9A01F02	AAS標準版 V10	更新國密配置說明
2025年3月	V10P9A01F01	AAS標準版 V10	新增整合鏈路追蹤工具說明
2024年9月	V10P8A01F02	AAS標準版 V10	新增與springboot整合部分內容 調整文中顯示格式等問題
2023年12月	V10P8A01F01	AAS標準版 V10	更新許可證授權碼獲取方式

3 產品概述

3.1 產品簡介

3.1.1 概述

金蝶 Apusic 應用伺服器V10標準版是一款支援 Jakarta EE 8 等價規範（javax 名稱空間）的web應用伺服器，支援全新Servlet和JSP規範，同時提供 jakarta 名稱空間應用的遷移部署能力。金蝶 Apusic 應用伺服器V10標準版技術先進、性能穩定、輕量簡便，快速響應隨需而變的業務需求，是應用系統高效開發的優選產品。

圖1：金蝶Apusic應用伺服器V10標準版架構圖



3.1.2 平台環境

金蝶Apusic應用伺服器V10標準版平台環境。

表1：金蝶Apusic應用伺服器V10標準版平台環境

作業系統	說明
Windows	Windows7、Windows10、Windows Server
Linux	CentOS、Ubuntu、SUSE、Redhat
國產平台	銀河麒麟系列、中標麒麟系列、普華、中科紅旗、深度等

3.1.3 系統環境

金蝶Apusic應用伺服器V10標準版系統環境。

表2：金蝶Apusic應用伺服器V10標準版系統環境

系統元件	系統要求
Java環境	JDK8及以上（相容JDK17），針對HTTP/2建議使用8u252以上
記憶體	至少需要 512MB 的記憶體，建議1024MB以上
磁碟空間	至少需要 1024MB 的磁碟空間，建議10G以上

3.1.4 JakartaEE規範

金蝶Apusic應用伺服器V10標準版產品規範（Jakarta EE 8 等價規範，javax 名稱空間）。

表3：金蝶Apusic應用伺服器V10標準版產品規範

規範/API	版本
Servlet	4.0
JSP	2.3
Expression Language	3.0
WebSocket	1.1

3.1.5 術語

AAS：金蝶Apusic應用伺服器

AAMS-[aas_version]：金蝶Apusic應用伺服器標準版產品標識

APUSIC_HOME：金蝶Apusic應用伺服器安裝目錄

3.2 產品清單

產品安裝包通常打包為 AAMS-[version].tar.gz 或 AAMS-[version].zip 。

文中 `${APUSIC_HOME}` 表示應用伺服器的安裝目錄。

表4：金蝶Apusic應用伺服器V10標準版產品包

產品包	說明
AAMS-[version].zip/AAMS-[version].tar.gz	金蝶Apusic應用伺服器V10標準版安裝包

金蝶Apusic應用伺服器V10標準版產品包說明。

表5：金蝶Apusic應用伺服器V10標準版產品包說明

文件	說明
bin	應用伺服器指令碼所在的目錄，除啟動（startapusic）、停止（stopapusic）、執行（apusic）指令碼外，還包括重啟（restartapusic）、版本檢視（version）、配置檢查（configtest）、密碼摘要（digest）、密碼套件（ciphers）、例項建立（makebase）、命令列工具（aascli）、安全基線（baseline）、調用鏈工具（apm/）等
conf	應用伺服器配置文件所在的目錄，包括 apusic.conf、context.xml、logging.properties、vm.options、安全基線配置（baseline/）等
console	管控平台程式所在的目錄
lib	應用伺服器JAR文件所在的目錄
logs	應用伺服器日誌文件默認目錄
plugins	應用伺服器外掛所在的目錄，包括 apollo、etcd、nacos、kafka、pinpoint、skywalking、redis、connReclaim 等外掛
store	應用伺服器持久化資料存放的目錄
webapps	部署應用程式包的目錄
webapps-console	管控平台內建應用的部署目錄
webapps-jakarta	jakarta 名稱空間應用的部署目錄
temp	應用伺服器臨時文件存放的目錄
work	存放JSP編譯後產生的class文件
license.xml	產品包的授權文件

change-log.md

產品變更日誌

4 安裝與執行

4.1 安裝與解除安裝

4.1.1 Java環境

安裝前，需要先確認java環境。支援JDK8及以上版本（相容JDK17），建議使用1.8.0_201及以上，檢視JDK版本命令如下：

```
[user@test bin]# java -version
java version "1.8.0_212"
Java(TM) SE Runtime Environment (build 1.8.0_212-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.212-b10, mixed mode)
```

支援指定 JAVA 執行環境：在 `${APUSIC_HOME}/bin` 目錄下建立 `setenv.sh`（Linux，需新增可執行許可權）或 `setenv.bat`（Windows）文件，啟動指令碼會自動載入，在文件中設置 `JAVA_HOME`，如：

```
JAVA_HOME="/opt/java/jdk1.8.0_271"
```

4.1.2 安裝

準備安裝包 `AAMS-V10.0.8-SP12-AE-20260701.tar.gz` 或 `AAMS-V10.0.8-SP12-AE-20260701.zip`，複製至有許可權的目錄，進行解壓。

解壓即安裝成功。

```
執行 tar -zxvf AAMS-V10.0.8-SP12-AE-20260701.tar.gz
或 unzip AAMS-V10.0.8-SP12-AE-20260701.zip
```

4.1.3 解除安裝

刪除安裝目錄 `${APUSIC_HOME}/` 下所有文件即可完成解除安裝操作。解除安裝將會丟失當前AAS所有已配置的文件，請謹慎操作。

4.2 許可證授權

AAS需要有對應的許可證才能正常使用，通常情況下，金蝶天燕會根據用戶購買的產品版本配套對應的許可證，如果在使用過程中出現許可證過期或無效等問題，建議優先聯絡對接的天燕服務人員，重新申請對應許可證。

產品授權方式分為普通授權和集中授權。

4.2.1 普通授權

普通授權指根據IP、域名等方式生成 `license.xml` 文件，將授權文件放置安裝目錄下，`${APUSIC_HOME}/license.xml`。



名称	大小	压缩后大小	类型
..			文件夹
work	0	0	文件夹
webapps-jakarta	0	0	文件夹
webapps-console	4,132,433	3,686,639	文件夹
webapps	0	0	文件夹
temp	0	0	文件夹
store	0	0	文件夹
plugins	66,442,713	61,212,681	文件夹
logs	0	0	文件夹
lib	28,538,926	25,838,490	文件夹
console	82,699,941	61,178,414	文件夹
conf	14,779,777	13,102,052	文件夹
bin	981,716	836,843	文件夹
license.xml	917	724	Microsoft Edge ...
change-log.md	3,863	1,725	Markdown File

如若安裝有管控平台，系統管理員admin登入管控平台，點選右上角的授權圖示，點選“更新授權”，上傳可用的授權文件，儲存即可完成更新。重啟AAS後新授權文件生效。

4.2.2 集中授權

集中授權指連線授權中心，進行統一授權。需要先搭建金蝶Apusic授權中心，操作方式可參考《金蝶Apusic許可授權中心用戶手冊》，或聯絡金蝶天燕技術支援人員。

在系統環境中配置環境變數，或在AAS安裝目錄根目錄建立 `acls.properties` 文件，新增以下引數：

```
apusic_acls_enable=true
apusic_acls_authUrls=127.0.0.1:6886
apusic_acls_ns=apusic
apusic_acls_tenant=ApusicTest
```

連線引數說明：

表6：金蝶Apusic許可授權中心連線引數說明

引數名	引數值說明
apusic_acls_enable	是否開啟授權中心認證，取值為true或false，為true則表示開啟授權中心認證。沒有該引數或該引數值為false，都表示沒有開啟授權中心認證；
apusic_acls_authUrls	授權中心的地址，可設置多個授權地址，格式為ip1:port1;ip2:port2，如果一個授權地址連結失敗，會輪詢其他的地址；如果開啟授權中心認證，則為必填引數，其中端口為授權中心的https端口；
apusic_acls_ns	設置該例項所屬的名稱空間名稱，可選引數；默認值為public，具體的名稱空間可以在授權中心管理控制台-系統管理-授權管理檢視。
apusic_acls_tenant	設置該例項所屬的租戶名稱，可選引數。

AAS啟動時將會自動連線到Apusic授權中心。

4.2.3 獲取特徵碼

如果在使用過程中出現許可證過期或無效等問題，建議優先聯絡對接的天燕服務人員，重新申請對應許可證。重新申請對應許可證時，需要將產品的特徵碼(auth code)提供到天燕對接人員。

在 `${APUSIC_HOME}/bin`，執行 `apusic.sh -ac [ethname or ip]`，`[ethname or ip]` 取值為ip地址或者網絡卡名稱，類似如下：

```
apusic.sh -ac 127.0.0.1
```

列印特徵碼資訊，類似如下，Auth Code=特徵碼內容：

```
Using APUSIC_BASE:    /opt/AAS/AAMS-V10.0.0
Using APUSIC_HOME:    /opt/AAS/AAMS-V10.0.0
Using APUSIC_TMPDIR:  /opt/AAS/AAMS-V10.0.0/temp
Using JRE_HOME:       /usr/local/java/jdk1.8.0_231/jre
Using CLASSPATH:      /opt/AAS/AAMS-V10.0.0/bin/bootstrap.jar:/opt/AAS/AAMS-V10.0.0/bin/aas-juli.jar
Using APUSIC_OPTS:
Auth Code=SZTY829390371
```

獲取特徵碼後再提供特徵碼申請授權文件。

4.2.4 嵌入式部署方式獲取特徵碼

在與springboot整合的方式下，沒有獲取特徵碼的 `apusic` 命令，需要依賴用戶的應用進行獲取。

如果已經有臨時的授權文件可以啟動應用，則先把授權文件移除，然後在啟動命令中增加JVM引數 `apusic-ac`，取值為ip地址或者網絡卡名稱，類似如下，eth3為測試機器的網絡卡名稱，需要根據實際的網路資訊修改：

```
java -Dapusic-ac=eth3 -jar boot-quickstart-1.0.jar
```

由於沒有授權文件，應用啟動會停止，在停止前會列印特徵碼資訊，類似如下，Auth Code=特徵碼內容：

```
Auth Code=SZTY263537626
license is invalid.
```

4.3 啟動與停止

4.3.1 在 Linux 環境下啟動

- 1、安裝完成並更新授權文件後，進入安裝目錄 `${APUSIC_HOME}/bin`，執行 `apusic.sh run`。
- 2、進入安裝目錄 `${APUSIC_HOME}/bin`，執行 `startapusic.sh`。
- 3、如果需要使用除錯模式（debug），可在啟動指令碼，`apusic.sh` 中設置 `JPDA_OPTS`，如：

```
JPDA_OPTS="-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000"
```

執行 `apusic.sh jpda start` 。

4.3.2 在 Windows 環境下啟動

- 安裝完成並更新授權文件後，進入安裝目錄 `${APUSIC_HOME}\bin`，執行 `apusic.bat run`。
- 進入安裝目錄 `${APUSIC_HOME}\bin`，執行 `startapusic.bat`。
- 如果需要使用除錯模式 (debug)，可在啟動指令碼，`apusic.bat` 中設置 `JPDA_OPTS`，如：

```
JPDA_OPTS=-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
```

執行 `apusic.bat jpda start` 。

建議優先檢查系統默認編碼是否為UTF-8。如果 Windows 啟動出現亂碼，則可通過修改 `conf/logging.properties` 文件，把控制台日誌編碼設置為 `GBK`。

```

1  # Handler specific properties.
2  # Describes specific configuration info for Handlers.
3  #####
4
5  1apusic.com.apusic.juli.AsyncFileHandler.level = FINE
6  1apusic.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
7  1apusic.com.apusic.juli.AsyncFileHandler.prefix = apusic.
8  1apusic.com.apusic.juli.AsyncFileHandler.maxDays = 90
9  1apusic.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
10
11
12  2localhost.com.apusic.juli.AsyncFileHandler.level = FINE
13  2localhost.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
14  2localhost.com.apusic.juli.AsyncFileHandler.prefix = localhost.
15  2localhost.com.apusic.juli.AsyncFileHandler.maxDays = 90
16  2localhost.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
17
18
19  3manager.com.apusic.juli.AsyncFileHandler.level = FINE
20  3manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
21  3manager.com.apusic.juli.AsyncFileHandler.prefix = manager.
22  3manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
23  3manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
24
25
26  4host-manager.com.apusic.juli.AsyncFileHandler.level = FINE
27  4host-manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
28  4host-manager.com.apusic.juli.AsyncFileHandler.prefix = host-manager.
29  4host-manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
30  4host-manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
31
32
33  java.util.logging.ConsoleHandler.level = FINE
34  java.util.logging.ConsoleHandler.formatter = com.apusic.juli.OnlineFormatter
35  java.util.logging.ConsoleHandler.encoding = GBK
36
37  #####
38  # Facility specific properties.
39  # Provides extra control for each logger.
40  #####
41
42  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].level = INFO
43  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].handlers = 2localhost.com.apusic.juli.AsyncFileHandler
44
45  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].level = INFO
46  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].handlers = 3manager.com.apusic.juli.AsyncFileHandler
47
48  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].level = INFO
49  com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].handlers = 4host-manager.com.apusic.juli.AsyncFileHandler
50
51
52
53

```

4.3.3 在 Linux 環境下停止

- 1、Ctrl + C

前端啟動狀態下，可以直接Ctrl + C退出，停止AAS。

2、stopapusic.sh

進入安裝目錄 `${APUSIC_HOME}/bin`，執行 `stopapusic.sh`。

3、kill程序

可以通過kill [程序號] 方式停止

4.3.4 在 Windows 環境下停止

1、Ctrl + C

前端啟動狀態下，可以直接Ctrl + C退出，停止AAS

2、stopapusic.bat

進入安裝目錄 `${APUSIC_HOME}\bin`，執行 `stopapusic.bat`。

3、停止程序

可以開啟工作管理員，找到對應程序，停止執行。

4.4 訪問管控平台

啟動時默認啟動管控平台，在瀏覽器中訪問系統地址：<https://127.0.0.1:6848/aas-console>。

使用系統管理員賬號登入：

- 用戶名稱： `admin`
- 密碼： `Apusic@123`

注意：首次登入強制修改密碼。

5 應用部署

5.1 自動部署

金蝶Apusic應用伺服器標準版提供應用部署功能，要部署應用，只需要把應用複製到 `${APUSIC_HOME}/webapps/` 目錄下即可自動部署。

5.2 指定路徑部署

因自動部署需要把應用複製到 `webapps` 下，現提供另一種方式部署：如果應用是壓縮包格式（war 格式等），需將應用解壓，比如解壓到 `D:/work_test/test-webapp`，然後編輯 `apusic.conf` 文件，在 `host` 下新增：

```
<application path="/test-webapp" docBase="D:\\work_test\\test-webapp" useHttpOnly="true" />
```

表7：金蝶Apusic應用伺服器V10標準版指定路徑部署配置說明

配置引數	說明	默認值
path	部署應用的上下文路徑	
docBase	解壓好的應用的位置，Windows環境下需要使用雙斜槓\\分級	
useHttpOnly	設置為true表示防止客戶端指令碼讀取JSESSIONID，防止CSRF/XSS惡意攻擊	

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <config>
3
4 <listeners>
5 <listener name="versionLoggerListener" className="com.apusic.ams.startup.VersionLoggerListener"/>
6 <listener name="aprilLifecycleListener" className="com.apusic.ams.core.AprilLifecycleListener" />
7 <listener name="jreMemoryLeakPreventionListener" className="com.apusic.ams.core.JreMemoryLeakPreventionListener"/>
8 <listener name="globalResourcesLifecycleListener" className="com.apusic.ams.beans.GlobalResourcesLifecycleListener"/>
9 <listener name="threadLocalLeakPreventionListener" className="com.apusic.ams.core.ThreadLocalLeakPreventionListener"/>
10 </listeners>
11
12 <resources>
13 <resource name="userDatabase" auth="Container" type="com.apusic.ams.UserDatabase"
14 description="User database that can be updated and saved"
15 factory="com.apusic.ams.users.MemoryUserDatabaseFactory"
16 pathName="conf/aas-users.xml" />
17 </resources>
18
19 <executors>
20 <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
21 prestartminSpareThreads="true" maxQueueSize="5000"/>
22 </executors>
23
24 <endpoints>
25 <endpoint name="ams-http" port="6888" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="6443" executor="http-thread-pool"/>
26 <endpoint name="ams-https" port="6443" protocol="HTTP/1.1"
27 maxThreads="150" SSLEnabled="true" >
28 <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
29 <SSLHostConfig>
30 <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
31 certificateFile="conf/localhost-rsa-cert.pem"
32 certificateChainFile="conf/localhost-rsa-chain.pem"
33 type="RSA" />
34 </SSLHostConfig>
35 </endpoint>
36 <endpoint name="ams-ajp" port="6009" protocol="AJP/1.3" redirectPort="6443" />
37 </endpoints>
38
39 <services>
40 <service name="ams-service" defaultHost="localhost">
41 <realm className="com.apusic.ams.realm.LockOutRealm">
42 <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
43 </realm>
44 <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
45 <valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="localhost_access_log" suffix=".txt" pattern="%h %l %u %t &quot;%r&quot; %s %b" />
46 <application path="/test-webapp" docBase="D:\\work_test\\test-webapp" useHttpOnly="true"/>
47 </host>
48 </service>
49 </services>
50
51 <servers>
52 <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
53 </servers>
54
55 <containers>
56 <container name="container" port="6005" shutdown="shutdown" servers="ams-server" resources="userDatabase"
57 listeners="versionLoggerListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener,aprilLifecycleListener"/>
58 </containers>
59 </config>

```

5.3 手動部署多個應用

在普通方式部署時，應用除了直接複製到 `webapps` 目錄自動進行部署外，也可以通過修改配置文件手動部署：修改 `apusic.conf` 文件中的 `host` 標籤，增加如下的類似配置，其中部署了 2 個 `application`，屬性 `path` 為指定的上下文路徑，`docBase` 是應用的目錄：

```

<host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
  <application path="/test1" docBase="H:/apps/test1.war" useHttpOnly="true" />
  <application path="/test2" docBase="H:/apps/test2.war" useHttpOnly="true" />
</host>

```

5.4 熱部署配置

應用伺服器默認支援熱部署。

如果想要關閉熱部署，在 `${APUSIC_HOME}/conf/vm.options` 文件中新增以下引數（每行一個），儲存後重啟應用伺服器，即可關閉熱部署功能（Windows 與 Linux 通用）。

```
-Dcom.apusic.onlyOneAutoDeploy.enabled=true
```

5.5 配置文根對映

文根對映指將訪問 URL 中的上下文路徑（context path）與伺服器上的物理部署目錄（docBase）進行繫結，實現“不同路徑 → 不同目錄”或“同一目錄 → 多個訪問路徑”的靈活對映。

配置方式：

修改 `apusic.conf` 中的 `ams-service` 配置 `host`，將 `unpackWARs` 設置為 `false`，定義 `application` 中的 `path` 值。

示例：

```
<host name="localhost" appBase="webapps" unpackWARs="false" autoDeploy="false" appBaseReDeploy="false"
deployOnStartup="false">
    <application path="/" useHttpOnly="true"/>
    <valve className="com.apusic.ams.valves.ErrorReportValve" showReport="false"
showServerInfo="false" />
    <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs"
prefix="access_log" suffix=".txt" pattern="%{yyyy-MM-dd HH:mm:ss}t %I %U %m %a %D %s"
rotatable="true" maxDays="3" buffered="false" />
</host>
```

5.6 取消部署

進入 `${APUSIC_HOME}/webapps/` 將對應的應用程式文件刪除即可完成解除安裝操作。

如果是指定路徑部署，停止執行應用伺服器，進入 `${APUSIC_HOME}/conf/` 編輯 `apusic.conf` 文件，刪除對應應用程式的 `application` 項，如：

```
<application path="/test-webapp" docBase="D:\\work_test\\test-webapp" useHttpOnly="true" />
```

6 配置與運維

6.1 配置文件總覽

apusic.conf 是金蝶 Apusic 應用伺服器的重要配置文件，可通過該文件設置端口、線程池大小、設置部署應用路徑、性能調優等等。

默認配置如：

```
<config>
  <listeners>
    <listener name="aprLifecycleListener" className="com.apusic.ams.core.AprLifecycleListener" />
    <listener name="jreMemoryLeakPreventionListener"
className="com.apusic.ams.core.JreMemoryLeakPreventionListener"/>
    <listener name="globalResourcesLifecycleListener"
className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener"/>
    <listener name="threadLocalLeakPreventionListener"
className="com.apusic.ams.core.ThreadLocalLeakPreventionListener"/>
  </listeners>
  <resources>
    <resource name="userDatabase" auth="Container" type="com.apusic.ams.UserDatabase"
      description="User database that can be updated and saved"
      factory="com.apusic.ams.users.MemoryUserDatabaseFactory"
      pathname="conf/aas-users.xml" />
  </resources>
  <executors>
    <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
      prestartminSpareThreads="true" maxQueueSize="5000"/>
  </executors>
  <endpoints>
    <endpoint name="ams-http"
      port="6888"
      protocol="HTTP/1.1"
      connectionTimeout="30000"
      redirectPort="6443"
      executor="http-thread-pool"
      maxConnections="10000"
      enableLookups="false"
      acceptCount="100"
      maxPostSize="104857600"
      maxHttpRequestSize="8192"
      compression="on"
      disableUploadTimeout="true"
      compressionMinSize="2048"

compressibleMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-
font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image/png,image/gif,audio/mpeg,video/mp4"
      URIEncoding="utf-8"
      processorCache="20000"
      tcpNoDelay="true"
      keepAliveTimeout="15000"
      maxKeepAliveRequests="100">
    </endpoint>
    <endpoint name="ams-jmx"
      port="6868"
```

```

        auth="false"
        protocol="JMXRMI/2.0">
    </endpoint>
</endpoints>
<services>
    <service name="ams-service" defaultHost="localhost">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
        </host>
    </service>
</services>
<servers>
    <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-
pool"/>
</servers>
<containers>
    <container name="container" port="-1" shutdown="shutdown" servers="ams-server"
resources="userDatabase"
listeners="aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLoca
</containers>
</config>

```

apusic.conf 各模組的完整引數說明見附錄：listeners 見附錄A、resources 見附錄B、executors 見附錄C、endpoints 見附錄D、services 與 servers 見附錄E、containers 見附錄F；數據源 Resource 引數見附錄G、Web 通用配置見附錄H。

6.2 設置 JVM 記憶體

JVM 引數通過 `${APUSIC_HOME}/conf/vm.options` 文件設置，啟動指令碼（`apusic.bat` / `apusic.sh`）啟動伺服器時會自動讀取該文件，Windows 與 Linux 通用。

在 `vm.options` 文件中每行填寫一個 JVM 引數（以 `#` 開頭的行為註釋），例如：

```

-Xms2g
-Xmx2g
-XX:NewRatio=2
-XX:MetaspaceSize=256m
-XX:MaxMetaspaceSize=256m

```

出廠默認配置為 `-Xms1024m`、`-Xmx4096m`。修改後需重啟應用伺服器生效。

注意： `stop`、`configtest`、`baseline` 等一次性命令不會讀取 `vm.options`，僅伺服器啟動時生效。

如需設置環境變數（如 `JAVA_HOME`、`JAVA_OPTS`），可在 `${APUSIC_HOME}/bin` 目錄下建立 `setenv.bat`（Windows）或 `setenv.sh`（Linux，需新增可執行許可權），啟動指令碼會自動載入。

6.3 配置數據源

6.3.1 通用配置

應用伺服器提供數據源配置功能。

1. 進入 `${APUSIC_HOME}/conf/`，編輯 `context.xml` 文件，新增資料庫連線池配置資訊，如下：

```

<Resource name="jdbc/test" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.cj.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&characterEncoding=utf-8"

```

```
username="root"
password="[YOUR_PASSWORD]" maxTotal="100" maxIdle="30" maxWaitMillis="10000" />
```

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- The contents of this file will be loaded for each web application -->
3  <Context>
4
5      <!-- Default set of monitored resources. If one of these changes, the    -->
6      <!-- web application will be reloaded.                                -->
7      <WatchedResource>WEB-INF/web.xml</WatchedResource>
8      <WatchedResource>WEB-INF/aas-web.xml</WatchedResource>
9      <WatchedResource>${apusic.base}/conf/web.xml</WatchedResource>
10
11     <!-- Uncomment this to disable session persistence across Aas restarts -->
12     <!--
13     <Manager pathname="" />
14     -->
15
16     <Resource name="jdbc/test"
17             auth="Container"
18             type="javax.sql.DataSource"
19             driverClassName="com.mysql.jdbc.Driver"
20             url="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&characterEncoding=utf-8"
21             username="root"
22             password="root"
23             maxTotal="100"
24             maxIdle="30"
25             maxWaitMillis="10000"/>
26
27 </Context>
28

```

表8：金蝶Apusic應用伺服器V10標準版數據源配置說明

配置引數	說明	值型別	默認值
name	數據源 jndi 名稱	字元	
auth	容器負責資源的連線	字元	
type	繫結型別	字元	
url	資料庫連線 url	字元	
username	用戶名稱	字元	
password	密碼	字元	
maxTotal	最大連線數	字元	
maxIdle	最大空閒連線數	字元	
maxWaitMillis	獲取連線的最大等待時間（單位毫秒），-1 表示無限等待	字元	-1
minIdle	最小空閒連線數	字元	
validationQuery	連線有效性檢測語句	字元	
validationQueryTimeout	檢測語句超時時間，-1表示不超時	字元	-1
testOnBorrow	true 或 false，從池中拿連線的時，是否檢測連線有效性	布林值	true
testOnReturn	true 或 false，還回連線到池裡面時，是否檢測連線有效性	布林值	false
removeAbandonedOnBorrow	true 或 false，是否移除失效的連線	布林值	false
logAbandoned	true 或 false，是否列印移除失效連線的資訊及洩漏程式碼堆疊	布林值	false
testWhileIdle	true 或 false，連線空閒時，是否檢測連線有效性	布林值	false
minEvictableIdleTimeMillis	連線空閒時間，超過該時間會從池中移除（單位為毫秒）	整型	
timeBetweenEvictionRunsMillis	每隔多少時間去檢測一次空閒連線是否超時，默認不檢測（單位為毫秒）	整型	
factory	當設置為“com.apusic.aas.jdbc.pool.DataSourceFactory”時，用於在JNDI數據源配置中，指定由Apusic自帶的連線池實現來建立 javax.sql.DataSource、javax.sql.XADataSource例項，而非第三方或容器默	字元	

	認連線池。 如若設置了factory="com.apusic.aas.jdbc.pool.DataSourceFactory"，引數maxTotal、maxWaitMillis、removeAbandonedOnBorrow設置無效，需要對應設置maxActive、maxWait、removeAbandoned		
removeAbandoned	true 或 false，是否開啟自動回收洩漏連線。通常用於設置了factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	布林值	false
removeAbandonedTimeout	連線超時時間，超過該值未歸還視為洩漏，通常`removeAbandoned=true`時才生效	整型	
threshold	定義超過該時間判定為耗時SQL，單位為毫秒（ms）	整型	
notifyPool	true 或 false，控制慢 SQL 觸發後，是否通知連線池核心做狀態聯動。為“true”時，慢查詢觸發時，通知連線池、標記連線、可參與池監控 / 回收邏輯；為“false”時，只打印慢查詢日誌，不通知連線池，無額外池邏輯開銷	布林值	false
logFailed	true 或 false，是否列印錯誤日誌	布林值	false
maxActive	最大活躍連線數，連線池中同一時刻能同時分配給業務使用的最大數據庫連線數量。通常用於設置了factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	整型	
maxWait	最大等待時間，當連線池已滿（達到 maxActive），新業務請求從連線池獲取空閒連線時，最長阻塞等待的時間，單位毫秒（ms）。通常用於設置了factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	整型	

2.將數據源對應版本的驅動放入到 `${APUSIC_HOME}/lib/` 目錄下。

注意：在程式碼中要引用jndi 需要加上字首：`java:comp/env`，如上面的數據源的：`java:comp/env/jdbc/test`。

6.3.2 開啟耗時SQL語句追蹤

修改配置文件 `${APUSIC_HOME}/conf/context.xml` 文件，新增資料庫連線池配置資訊，配置 `jdbcInterceptors`。

- `threshold`：定義超過該時間判定為耗時SQL，單位為毫秒（ms）。
- `logFailed`：是否輸出錯誤日誌。
- `notifyPool`：控制慢 SQL 觸發後，是否通知連線池核心做狀態聯動。為“true”時，慢查詢觸發時，通知連線池、標記連線、可參與池監控 / 回收邏輯；為“false”時，只打印慢查詢日誌，不通知連線池，無額外池邏輯開銷。

配置如下：

```
<Resource auth="Container"
    factory="com.apusic.aas.jdbc.pool.DataSourceFactory"
    driverClassName="com.mysql.cj.jdbc.Driver"

    jdbcInterceptors="com.apusic.aas.jdbc.pool.interceptor.SlowQueryReportJmx(threshold=100,logFailed=true,notifyPool=true)"
    maxIdle="30" maxTotal="100" maxWaitMillis="10000"
    name="jdbc/TestDB"
    password="[YOUR_PASSWORD]"
    type="javax.sql.DataSource"
    url="jdbc:mysql://127.0.0.1:3306/test?serverTimezone=UTC"
    username="*****"/>
```

6.3.3 配置連線洩漏回收

連線洩漏：指應用從連線池（如資料庫連線池、HTTP 連線池）獲取連線後，因程式碼異常、邏輯疏漏未執行 `close()` 歸還，導致連線被長期佔用、無法複用的資源洩漏問題。

連線洩漏回收：連線池通過自動檢測 + 強制回收機制，釋放長期未歸還的洩漏連線，避免連線池資源耗盡、服務雪崩。

應用伺服器提供 `connectionAutoReclaim` 和 `removeAbandoned` 配置：

- `connectionAutoReclaim`：表示在請求處理過程中判斷，請求獲取資料庫連線時儲存連線，請求結束時主動檢查請求是否回收連線。`true` 為啟用；`false` 為停用。配置該值需要同時在 `executors` 連線池配置回收方式。

- `removeAbandoned`：通過空閒時間是否超過設置的超時時間判斷，是連線池的功能。`true` 為開啟自動回收（超時未釋放的連線將被強制回收）；`false` 為不回收（默認值）。

6.3.3.1 connectionAutoReclaim配置

1、修改配置文件 `${APUSIC_HOME}/conf/context.xml` 文件，新增資料庫連線池配置資訊。

- `connectionAutoReclaim`：是否開啟自動回收資料庫洩漏連線，表示在請求處理過程中判斷，請求獲取資料庫連線時儲存連線，請求結束時主動檢查請求是否回收連線。`true` 為啟用；`false` 為停用。配置該值需要同時在 `executors` 連線池配置回收方式。

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.cj.jdbc.Driver" url="jdbc:mysql://127.0.0.1:3306/test?serverTimezone=UTC"
username="mysqluser" password="*****" maxTotal="100" maxIdle="100" maxWaitMillis="3000"
connectionAutoReclaim="true"
logAbandoned="true" />
```

```
<Resource name="jdbc/TestDB"
auth="Container"
factory="com.apusic.aas.jdbc.pool.DataSourceFactory"
type="javax.sql.DataSource"
driverClassName="com.mysql.cj.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/user?useSSL=false&useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC"
username="mysqluser"
password="*****"
maxActive="20"
initialSize="20"
maxIdle="10"
minIdle="4"
maxWait="15000"
validationQuery="SELECT SLEEP(0.1)"
validationQueryTimeout="5"
testOnBorrow="true"
testOnReturn="true"
testWhileIdle="true"
testOnConnect="true"
validationInterval="3000"
jdbcInterceptors="com.apusic.aas.jdbc.pool.interceptor.SlowQueryReportJmx(threshold=100,logFailed=true,notifyPool=false)"
connectionAutoReclaim="true"
timeBetweenEvictionRunsMillis="60000"
minEvictableIdleTimeMillis="60000"
maxAge="900000"
numTestsPerEvictionRun="3"
removeAbandoned="false"
removeAbandonedTimeout="30000"
defaultAutoCommit="true"
logAbandoned="false"/>
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!--
<Manager pathname="" />
```

2、修改配置文件 `${APUSIC_HOME}/conf/apusic.conf` 修改 `executors` 連線池配置。

- `connectionAutoReclaim`：配置自動回收資料庫洩漏連線方式。`false` 為停用，表示檢測有洩漏的連線後，該連線將無法使用，但未被回收；`detect` 為只檢測，表示檢測有洩漏的連線，只顯示洩漏狀態，該連線仍可使用；`reclaim` 為檢測並回收，表示檢測到有洩漏的連線後，自動回收該洩漏連線。

```
<executors>
  <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
    prestartminSpareThreads="true" maxQueueSize="5000" connectionAutoReclaim="detect"/>
</executors>
```

```
<executors>
  <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
    prestartminSpareThreads="true" maxQueueSize="5000" connectionAutoReclaim="detect"
    threadStepReclaim="false" threadStepSize="20" threadStepTime="10000"/>
</executors>
```

6.3.3.2 removeAbandoned配置

1、修改配置文件 `${APUSIC_HOME}/conf/context.xml` 文件，新增資料庫連線池配置資訊。

- `removeAbandoned`：是否自動回收洩漏的連線，通過空閒時間是否超過設置的超時時間判斷，是連線池的功能。`true` 為開啟自動回收（超時未釋放的連線將被強制回收）；`false` 為不回收（默認值）。
- `removeAbandonedTimeout`：連線洩漏超時時間，單位為秒。例如設置值為60，一個連線被拿走後超過 60 秒未歸還視為洩漏；`removeAbandoned=true` 時才生效。
- `logAbandoned`：是否列印連線洩漏日誌記錄。

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.cj.jdbc.Driver" url="jdbc:mysql://127.0.0.1:3306/test?serverTimezone=UTC"
username="mysqluser" password="*****" maxTotal="100" maxIdle="100" maxWaitMillis="3000"
removeAbandoned="true"
removeAbandonedTimeout="60"
logAbandoned="true" />
```

```
-->
<!-- The contents of this file will be loaded for each web application -->
<Context>

<!-- Default set of monitored resources. If one of these changes, the -->
<!-- web application will be reloaded. -->
<WatchedResource>WEB-INF/web.xml</WatchedResource>
<WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
<WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
<!-- MySQL Database DataSource -->
<Resource name="jdbc/TestDB"
auth="Container"
type="javax.sql.DataSource"
factory="com.apusic.aas.jdbc.pool.DataSourceFactory"
driverClassName="com.mysql.cj.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC"
username="mysqluser"
password="*****"
maxActive="10000"
initialSize="10"
maxIdle="100"
minIdle="20"
maxWait="15000"
connectionAutoReclaim="true"
validationQuery="SELECT 1"
validationQueryTimeout="1"
testOnBorrow="true"
testOnReturn="true"
testWhileIdle="true"
testOnConnect="true"
timeBetweenEvictionRunsMillis="10000"
minEvictableIdleTimeMillis="60000"
numTestsPerEvictionRun="5"
removeAbandoned="false"
removeAbandonedOnMaintenance="false"
removeAbandonedTimeout="30000"
logAbandoned="true"
abandonWhenPercentageFull="10"
defaultAutoCommit="true"/>
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!--
sessionPersistence="none" />
```

6.3.3.3 配置手動回收洩漏連線

應用伺服器提供手動回收洩漏連線工具，當檢測到有洩漏連線，且未開啟自動回收洩漏連線功能時，可通過該工具手動回收洩漏連線。

配置方式：

1、修改配置文件 `${APUSIC_HOME}/conf/context.xml` 文件，新增資料庫連線池配置資訊。

- `connectionAutoReclaim`：是否開啟自動回收資料庫洩漏連線，表示在請求處理過程中判斷，請求獲取資料庫連線時儲存連線，請求結束時主動檢查請求是否回收連線。`true` 為啟用；`false` 為停用（默認值），如要手動回收洩漏連線，需要設置為 `true`。配置該值需要同時在 `apusic.conf` 中的 `executors` 連線池配置回收方式。
- `removeAbandoned`：是否自動開啟回收洩漏的連線，通過空閒時間是否超過設置的超時時間判斷，是連線池的功能。`true` 為開啟自動回收（超時未釋放的連線將被強制回收）；`false` 為不回收（默認值），如要手動回收洩漏連線，需要設置為 `false`。
- `removeAbandonedTimeout`：連線洩漏超時時間，單位為秒。例如設置值為60，一個連線被拿走後超過 60 秒未歸還視為洩漏；`removeAbandoned=true` 時才生效。
- `logAbandoned`：是否列印連線洩漏日誌。

配置示例如下：

```
<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
driverClassName="com.mysql.cj.jdbc.Driver" url="jdbc:mysql://127.0.0.1:3306/test?serverTimezone=UTC"
username="mysqluser" password="*****" maxTotal="100" maxIdle="100" maxWaitMillis="3000"
removeAbandoned="false"
removeAbandonedTimeout="60"
logAbandoned="true" />
```

```

-->
<!-- The contents of this file will be loaded for each web application -->
<Context>

<!-- Default set of monitored resources. If one of these changes, the -->
<!-- web application will be reloaded. -->
<WatchedResource>WEB-INF/web.xml</WatchedResource>
<WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
<WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
<!-- MySQL Database DataSource -->
<Resource name="jdbc/TestDB"
  auth="Container"
  type="javax.sql.DataSource"
  factory="com.apusic.aas.jdbc.pool.DataSourceFactory"
  driverClassName="com.mysql.cj.jdbc.Driver"
  url="jdbc:mysql://:3306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC"
  username="root"
  password="root"
  maxActive="10000"
  initialSize="10"
  maxIdle="100"
  minIdle="20"
  maxWait="15000"
  connectionAutoReclaim="true"
  validationQuery="SELECT 1"
  validationQueryTimeout="1"
  testOnBorrow="true"
  testOnReturn="true"
  testWhileIdle="true"
  testOnConnect="true"
  timeBetweenEvictionRunsMillis="10000"
  minEvictableIdleTimeMillis="60000"
  numTestsPerEvictionRun="5"
  removeAbandoned="false"
  removeAbandonedOnMaintenance="false"
  removeAbandonedTimeout="30000"
  logAbandoned="true"
  abandonWhenPercentageFull="10"
  defaultAutoCommit="true"/>
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!--
<SessionManager>org.apache.catalina.session.StandardSessionManager</SessionManager>
-->

```

如果開啟 `connectionAutoReclaim` 配置，還要修改配置文件 `${APUSIC_HOME}/conf/apusic.conf` 修改 `executors` 連線池配置。

- `connectionAutoReclaim` :配置自動回收資料庫洩漏連線方式。 `false` 為停用，表示檢測有洩漏的連線後，該連線將無法使用，但未被回收； `detect` 為只檢測，表示檢測有洩漏的連線，只顯示洩漏狀態，該連線仍可使用； `reclaim` 為檢測並回收，表示檢測到有洩漏的連線後，自動回收該洩漏連線。

```

<executors>
  <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
    prestartminSpareThreads="true" maxQueueSize="5000" connectionAutoReclaim="detect"/>
</executors>

```

```

<executors>
  <executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
    prestartminSpareThreads="true" maxQueueSize="5000" connectionAutoReclaim="detect"
    threadStepReclaim="false" threadStepSize="20" threadStepTime="10000"/>
</executors>

```

2、修改 `apusic.conf` 開啟 `ams-jmx`：將 `ams-jmx` 加入 `server` 模組的 `endpoints` 屬性中（如 `endpoints="ams-http,ams-jmx"`），詳細步驟見上文「開啟 JMX」。

3、進入 `${APUSIC_HOME}/plugins/connReclaim` 目錄執行以下命令，檢測洩漏連線。

```
java -jar connection-reclaim-1.0.1.jar
```

4、檢測到有洩漏的連線，執行 `close ID`，`ID` 為具體值，手動關閉洩漏連線。

```
close ID
```

```

+ 按上箭头键查看历史命令
> refresh
连接信息展示:
应用: /sqlDemo
数据源: jdbc/TestDB

```

连接ID	数据源JNDI名称	创建时间	上一次使用时间	状态	是否泄露
7	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
3	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露
5	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
2	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露
8	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
6	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
4	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露
0	jdbc/TestDB	2026-05-09 17:35:12	2026-05-09 17:35:25	活跃	疑似泄露
1	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露

```

=====
总计: 1 个应用, 9 个连接
=====

===== 命令操作 =====
请使用以下命令:
- 'refresh' 刷新列表
- 'return [NO.]' 归还指定连接
- 'close [NO.]' 关闭指定连接
- 'return [NO.,NO,...]' 归还多个连接
- 'close [NO.,NO,...]' 关闭多个连接
- 'return leaked' 归还所有泄漏连接
- 'close leaked' 关闭所有泄漏连接
- 'return all' 归还所有连接
- 'close all' 关闭所有连接
- 'exit' 退出程序
- 按上箭头键查看历史命令

> close 1
将关闭 1 个连接:
数据源=jdbc/TestDB, 连接ID=1, 状态=活跃
成功关闭, 数据源=jdbc/TestDB, 连接ID=1

操作完成: 成功 1 个, 失败 0 个

连接信息展示:
应用: /sqlDemo
数据源: jdbc/TestDB

```

连接ID	数据源JNDI名称	创建时间	上一次使用时间	状态	是否泄露
7	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
3	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露
5	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
2	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露
8	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
6	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:45	活跃	疑似泄露
4	jdbc/TestDB	2026-05-09 17:35:11	2026-05-09 17:36:44	活跃	疑似泄露

6.4 日誌配置

6.4.1 系統日誌配置

通過配置系統日誌文件配置日誌資訊，日誌文件為 `logging.properties`。

內部採用 JULI（Apache Commons Logging 的定製分支）並強制繫結 `java.util.logging`，以確保伺服器與 Web 應用的日誌系統相互隔離；若需更換日誌框架，則需配置重定向並確保新框架支援多類載入器環境。

```
handlers = 1apusic.com.apusic.juli.AsyncFileHandler, 2localhost.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler
.handlers = 1apusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler

#####
# Handler specific properties.
# Describes specific configuration info for Handlers.
#####

1apusic.com.apusic.juli.AsyncFileHandler.level = FINE
1apusic.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
1apusic.com.apusic.juli.AsyncFileHandler.prefix = apusic.
1apusic.com.apusic.juli.AsyncFileHandler.maxDays = 90
1apusic.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

2localhost.com.apusic.juli.AsyncFileHandler.level = FINE
2localhost.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
2localhost.com.apusic.juli.AsyncFileHandler.prefix = localhost.
2localhost.com.apusic.juli.AsyncFileHandler.maxDays = 90
2localhost.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

com.apusic.juli.logging.ApusicConsoleHandler.level = FINE
com.apusic.juli.logging.ApusicConsoleHandler.formatter = com.apusic.juli.OneLineFormatter
com.apusic.juli.logging.ApusicConsoleHandler.encoding = UTF-8

#####
# Facility specific properties.
# Provides extra control for each logger.
#####

com.apusic.ams.core.ContainerBase.[ams-service].[localhost].level = INFO
com.apusic.ams.core.ContainerBase.[ams-service].[localhost].handlers = 2localhost.com.apusic.juli.AsyncFileHandler

# For example, set the com.apusic.ams.util.LifecycleBase logger to log
# each component that extends LifecycleBase changing state:
#com.apusic.ams.util.LifecycleBase.level = FINE

# To see debug messages in TldLocationsCache, uncomment the following line:
#com.apusic.jasper.compiler.TldLocationsCache.level = FINE

# To see debug messages for HTTP/2 handling, uncomment the following line:
#com.apusic.connector.http2.level = FINE

# To see debug messages for WebSocket handling, uncomment the following line:
#com.apusic.aas.websocket.level = FINE
```

表9: 金蝶Apusic應用伺服器V10.0.8標準版 logging.properties 日誌配置說明

配置引數	說明	默認值
handlers	根日誌處理器列表，所有日誌默認會交給這些處理器處理	1apusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler
.handlers	默認類/包級別的處理器，未單獨配置的日誌記錄器（Logger）會繼承此配置	1apusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler

表10: 金蝶Apusic應用伺服器V10.0.8標準版 logging.properties 處理程序特定屬性

配置引數	說明	默認值
1apusic.com.apusic.juli.AsyncFileHandler.level	設置該日誌處理器的日誌級別，可設置為ALL、FINEST、FINER、FINE、INFO、WARNING、SEVERE	FINE
1apusic.com.apusic.juli.AsyncFileHandler.directory	指定日誌文件的儲存目錄	\${apusic.base}/logs
1apusic.com.apusic.juli.AsyncFileHandler.prefix	設置日誌文件的檔名字首	apusic.
1apusic.com.apusic.juli.AsyncFileHandler.maxDays	設置日誌文件的最大保留天數	90
1apusic.com.apusic.juli.AsyncFileHandler.encoding	設置日誌文件的字元編碼	UTF-8
1apusic.com.apusic.juli.AsyncFileHandler.fileTimestampFormat	設置日誌文件按日期輪轉的時間戳格式，例如 yyyy-MM-dd-HH-mm-ss-SSS	yyyy-MM-dd
1apusic.com.apusic.juli.AsyncFileHandler.filePattern	設置日誌文件單行日誌的輸出格式	%d{yyyy-MM-dd HH:mm:ss.SSS} %level [%t] %logger %msg%n
com.apusic.juli.logging.ApusicConsoleHandler.level	設置控制台輸出的日誌級別	FINE
com.apusic.juli.logging.ApusicConsoleHandler.formatter	設置控制台輸出的日誌單行格式	com.apusic.juli.OneLineFormatter

com.apusic.juli.logging.ApusicConsoleHandler.encoding	設置控制台輸出的日誌編碼	UTF-8
com.apusic.juli.logging.ApusicConsoleHandler.enabled	是否啟用控制台日誌輸出	true
com.apusic.juli.logging.ApusicConsoleHandler.pattern	設置控制台單行日誌的輸出格式	%d{yyyy-MM-dd HH:mm:ss.SSS} %level [%t] %logger %msg%n

表11: 金蝶Apusic應用伺服器V10.0.8標準版 logging.properties 特定服務日誌配置

說明：以下為可選配置示例，出廠 logging.properties 中默認未配置。如需使用 2localhost 處理器實現服務日誌隔離，需先在文件中定義該處理器（參考 1apusic 處理器的配置方式）並加入 handlers 列表。

配置引數	說明	默認值
com.apusic.ams.core.ContainerBase.[ams-service].[localhost].level	ams-server服務的日誌級別 com.apusic.ams.core.ContainerBase:日誌記錄器的類名，對應 AMS 服務的容器基礎類 [ams-service]:服務名稱，限定日誌作用於 ams-service 服務 [localhost]: 虛擬主機名稱，限定日誌作用於 localhost 虛擬主機 level = INFO: 該服務的日誌級別為 INFO，只輸出 INFO 及以上級別（WARNING/SEVERE）的日誌，過濾掉除錯資訊	INFO
com.apusic.ams.core.ContainerBase.[ams-service].[localhost].handlers	該服務的日誌只輸出到 2localhost 非同步文件處理器（對應 localhost.log），不會輸出到全域性 apusic.log 或控制台，實現服務日誌隔離	2localhost.com.apusic.juli.AsyncFileHandler

表12: 金蝶Apusic應用伺服器V10.0.8標準版 logging.properties 其他配置

說明：以下為可選配置示例，出廠 logging.properties 中默認未配置，按需新增。

配置引數	說明	默認值
com.apusic.ams.util.LifecycleBase.level	開啟後會記錄所有繼承 LifecycleBase 的元件狀態變化（如啟動、停止、初始化），級別為 FINE（除錯級）	FINE
com.apusic.connector.http2.level	開啟後會記錄 HTTP/2 協議處理的詳細除錯資訊	FINE
com.apusic.aas.websocket.level	開啟後會記錄 WebSocket 連線、訊息處理的詳細除錯資訊	FINE

6.4.2 配置訪問日誌

訪問日誌可配置在 apusic.conf 中的 application 或者 service 裡面。配置在 host 裡面則該 host 下的應用訪問都會被記錄日誌。出廠默認未配置訪問日誌（AccessLogValve），需要時按以下方式新增。

```
<Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log"
suffix=".txt"
pattern="common" />

</endpoints>

<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="common"/>
      <application path="/test" docBase="E:\work test\test" useHttpOnly="true">
        <!-- <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="common"/> -->
      </application>
    </host>
  </!-->
</services>
```

表13: 金蝶Apusic應用伺服器V10標準版訪問日誌配置說明

配置引數	說明	值型別	默認值
------	----	-----	-----

directory	日誌儲存目錄	字元	
prefix	日誌文件字首	字元	
suffix	日誌文件字尾	字元	
pattern	日誌文件格式	字元	common
rotatable	是否開啟日誌自動切割，按天滾動	布林值	true
maxDays	設置訪問日誌保留時長，單位為天	字元	
buffered	開啟日誌快取，為true時表示先寫緩衝區，滿了再批次寫入，為false表示不快取，每條日誌立即寫入	布林值	true

pattern參考下面內容:

- %a: 遠端 IP 地址
- %A: 本地 IP 地址
- %b: 傳送的位元組數，不包括 HTTP 頭，或“-”如果沒有傳送位元組
- %B: 傳送的位元組數，不包括 HTTP 頭
- %h: 遠端主機名
- %H: 請求協議
- %l (小寫的 l): 遠端的用戶名稱
- %m: 請求方法
- %p: 本地端口
- %q: 查詢字串
- %r: 第一行的請求
- %s: 響應的 HTTP 狀態程式碼
- %S: 用戶會話 ID
- %t: 日期和時間，在通用日誌格式
- %{yyyy-MM-dd HH:mm:ss}t 輸出格式化的請求時間，年 - 月 - 日 時: 分: 秒
- %u: 遠端用戶身份驗證
- %U: 請求的 URL 路徑
- %v: 本地伺服器名
- %D: 處理請求的時間（以毫秒為單位）
- %T: 處理請求的時間（以秒為單位）
- %I(大寫的 I): 當前請求的線程名稱

此外，可以指定以下別名來設置：

common:%h %l %u %t "%r" %s %b

combined:%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-Agent}i"

另外，還可以將 request 請求的查詢引數、Session 會話變數值、Cookie 值或 HTTP 請求 / 響應頭內容的變數值等內容寫入到日誌文件。類似 Apache 的語法：

- %{XXX}i xxx 代表傳入的頭 (HTTP Request)
- %{XXX}o xxx 代表傳出的響應頭 (Http Response)
- %{XXX}c xxx 代表特定的 Cookie 名
- %{XXX}r xxx 代表 ServletRequest 屬性名
- %{XXX}s xxx 代表 HttpSession 中的屬性名

6.4.3 配置日誌推送

應用伺服器支援配置日誌推送到Kafka。配置方式如下：

6.4.3.1 類路徑修改

Windows環境下:

在啟動指令碼 apusic.bat 中 set "CLASSPATH=%CLASSPATH%;%APUSIC_BASE%\bin\aa-juli.jar" 下增加

```
set "CLASSPATH=%CLASSPATH%;%APUSIC_BASE%\plugins\kafka\*" :
```

```
if not exist "%APUSIC_BASE%\bin\aa-juli.jar" goto juliClasspathHome
set "CLASSPATH=%CLASSPATH%;%APUSIC_BASE%\bin\aa-juli.jar"
```

```
set "CLASSPATH=%CLASSPATH%;%APUSIC_BASE%\plugins\kafka\*"

```

Linux環境下:

在啟動指令碼 apusic.sh 中 CLASSPATH=\$CLASSPATH:\$APUSIC_BASE/bin/aas-juli.jar 下增加

```
CLASSPATH=$CLASSPATH:$APUSIC_BASE/plugins/kafka/* :

```

```
if [ -r "$APUSIC_BASE/bin/aas-juli.jar" ] ; then
    CLASSPATH=$CLASSPATH:$APUSIC_BASE/bin/aas-juli.jar
    CLASSPATH=$CLASSPATH:$APUSIC_BASE/plugins/kafka/*

```

6.4.3.2 配置

在 \${APUSIC_HOME}/conf/logging.properties 的 handlers 和 .handlers 增加處理類的配置:

```
handlers = lapusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler,
com.apusic.juli.logging.ApusicKafkaHandler

.handlers = lapusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler,
com.apusic.juli.logging.ApusicKafkaHandler

```

在配置文件下面空白處增加處理類引數配置:

```
com.apusic.juli.logging.ApusicKafkaHandler.bootstrapServers=localhost:9092
com.apusic.juli.logging.ApusicKafkaHandler.formatter=com.apusic.juli.OneLineFormatter
com.apusic.juli.logging.ApusicKafkaHandler.topic=log
com.apusic.juli.logging.ApusicKafkaHandler.clientId=aas
com.apusic.juli.logging.ApusicKafkaHandler.compressionType=none
com.apusic.juli.logging.ApusicKafkaHandler.maxBlockMs=10000
com.apusic.juli.logging.ApusicKafkaHandler.retries=1
com.apusic.juli.logging.ApusicKafkaHandler.requiredNumAcks=0
com.apusic.juli.logging.ApusicKafkaHandler.lingerMs=0
com.apusic.juli.logging.ApusicKafkaHandler.bufferMemory=33554432
com.apusic.juli.logging.ApusicKafkaHandler.batchSize=16384

```

com.apusic.juli.logging.ApusicKafkaHandler 引數說明:

表14: 金蝶Apusic應用伺服器V10日誌推送到Kafka配置說明

com.apusic.juli.logging.ApusicKafkaHandler.bootstrapServers	推送的Kafka叢集地址，可配置多個，格式:host1:port1,host2:port2...
com.apusic.juli.logging.ApusicKafkaHandler.formatter	推送到Kafka的日誌格式，支援自帶的com.apusic.juli.OneLineFormatter、com.apusic.juli.JdkLoggerFormatter、com.apusic.juli.VerbatimFormatter，可以通過繼承java.util.logging.Formatter支援自定義格式
com.apusic.juli.logging.ApusicKafkaHandler.topic	推送的Kafka 訂閱的主題名稱
com.apusic.juli.logging.ApusicKafkaHandler.clientId	請求時傳遞給伺服器的id字串。這樣做的目的是通過允許在伺服器端請求日誌中包含邏輯應用程式名稱，從而能夠跟蹤ip/端口之外的請求源
com.apusic.juli.logging.ApusicKafkaHandler.compressionType	指定給定主題的最終壓縮型別。“gzip”、“snappy”、“lz4”、“zstd”、以及none即不進行壓縮
com.apusic.juli.logging.ApusicKafkaHandler.maxBlockMs	訊息推送異常或快取記憶體滿時進行阻塞時間
com.apusic.juli.logging.ApusicKafkaHandler.retries	訊息推送失敗重試次數
com.apusic.juli.logging.ApusicKafkaHandler.requiredNumAcks	生產者在考慮完成請求之前要求leader收到的確認的數量。這控制了傳送的記錄的永續性。允許以下設置: requiredNumAcks=0: 設置為0，則生產者將完全不等待來自伺服器的任何確認。記錄將立即新增到socket緩衝區，並被認為已傳送。在這種情況下，不能保證伺服器已經收到記錄，重試配置將不會生效(因為客戶機通常不會知道任何失敗)。每個記錄返回的偏移量總是-1 requiredNumAcks=1: leader會將記錄寫到本地日誌中，但不會等待所有follower的完全確認。

	在這種情況下，如果leader在記錄失敗後立即失敗，但在追隨者複製記錄之前失敗，那麼記錄就會丟失 requiredNumAcks=all / -1:leader將等待完整的同步副本來確認記錄。這保證了只要至少有一個同步副本仍然存在，記錄就不會丟失。這是最有力的保證。這相當於requiredNumAcks=-1設置
com.apusic.juli.logging.ApusicKafkaHandler.lingerMs	延遲傳送時間：為減少負載和客戶端的請求數量，生產者不會一條一條傳送，而是會逗留一段時間批次傳送，當快取區的資料達到lingerMs時間或資料大小超過下列的batchSize就會進去推送
com.apusic.juli.logging.ApusicKafkaHandler.bufferMemory	生產者可以用來緩衝等待發送到伺服器的記錄的總記憶體位元組。如果記錄被髮送的速度超過了它們可以被髮送到伺服器的速度，那麼生產者將阻塞max.block。然後它會丟擲一個異常。默認32M
com.apusic.juli.logging.ApusicKafkaHandler.batchSize	傳送到緩衝區中的訊息會被分為一個一個的batch，分批次的傳送到broker 端，該引數就表示batch批次大小，默認16K

6.5 端口修改

修改 `apusic.conf` 文件中 `endpointname="ams-http"` 的 `port` 屬性，默認為 `port="6888"`。

6.6 開啟 JMX

JMX 服務默認為不開啟，如果需要開啟 JMX 服務，首先需要檢查配置文件中是否存在 `ams-jmx` 服務配置，如下：

```
<endpoint name="ams-jmx" port="6868" auth="false" protocol="JMXRMI/2.0">
</endpoint>
```

常用配置引數：

配置引數	說明	值型別	默認值
name	定義JMX協議名稱，例如 <code>ams-jmx</code>	字元	
port	JMX的偵聽端口	整型	6868
hostName	JMX的監聽地址	字元	localhost
protocol	JMX服務的協議	字元	JMXRMI/2.0
auth	是否啟用SSL安全連線，為true時需要配置授權認證	字元	false

如果沒有則需要新增上。再需要在 `apusic.conf` 中找到如下的配置，把 `ams-jmx` 增加到 `endpoints` 屬性中，如下配置：

```
<server name="ams-server" services="ams-service" endpoints="ams-http,ams-jmx" executors="http-thread-pool" />
```

JMX 默認訪問不進行授權驗證，如果需要授權認證，則需要修改相關配置並制定授權文件：

修改 `ams-jmx` 服務屬性 `auth` 設置為 `true`，配置密碼和訪問授權文件屬性 `passFile` 和 `accessFile`，默認為 `conf` 目錄下的 `jmxremote.password` 和 `jmxremote.access` 文件。

`jmxremote.password` 文件內容類似：

```
controlRole [YOUR_PASSWORD]
```

其中 `controlRole` 為用戶名稱，`[YOUR_PASSWORD]` 為密碼。

`jmxremote.access` 文件內容類似：

```
controlRole readonly
```

```
controlRole readwrite \
    create javax.management.monitor.* , javax.management.timer.* \
    unregister
```

6.7 開啟優雅停機

支援開啟優雅停機，AAS停止後，等待正在進行中的客戶端請求處理完畢。

在 \${APUSIC_HOME}/conf/apusic.conf 文件 endpoint 模組中新增 bindOnInit="false"；在 servers 模組新增配置 gracefulStopAwaitMillis="30000"，並重啟應用伺服器。

- bindOnInit：是否在伺服器啟動時自動將線程池繫結到對應服務 / 通道，這裡設置為false。
- gracefulStopAwaitMillis：AAS停止後，等待正在進行中的客戶端請求處理完畢的時長。單位毫秒，為0時表示關閉此功能。

```
<executor name="http-thread-pool" namePrefix="http-exec-" maxThreads="200" minSpareThreads="25"
    prestartminSpareThreads="true" maxQueueSize="5000"/>
</executors>
<endpoints>
  <endpoint name="ams-http"
    port="8089"
    protocol="HTTP/1.1"
    connectionTimeout="3000"
    redirectPort="6443"
    executor="http-thread-pool"
    maxConnections="10000"
    enableLookups="false"
    acceptCount="100"
    maxPostSize="104857600"
    maxHttpRequestSize="8192"
    compression="on"
    disableUploadTimeout="true"
    compressionMinSize="2048"
    compressibleMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/xml,application/javascript"
    audio/mpeg,video/mp4"
    URIEncoding="utf-8"
    processorCache="20000"
    tcpNoDelay="true"
    keepAliveTimeout="15000"
    bindOnInit="false"
    maxKeepAliveRequests="100">
  </endpoint>
  <endpoint name="ams-jmx"
    port="6868"
    auth="false"
    protocol="JMXRMI/2.0">
  </endpoint>
  <endpoint name="ams-monitor"
    port="18080"
    protocol="HTTP/1.1">
  </endpoint>
</endpoints>
```

```
</realm>
<host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
  <valve className="com.apusic.ams.valves.ErrorReportValve" showReport="false" showServerInfo="false" />
  <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="%{yyyy-MM-dd HH:mm:ss}t %I
ue" maxDays="3"/>
</host>
</service>
<service name="ams-monitor-service" defaultHost="localhost">
  <host name="localhost" appBase="monitorapps" unpackWARs="true" autoDeploy="true">
    <valve className="com.apusic.ams.valves.ErrorReportValve" showReport="false" showServerInfo="false" />
  </host>
</service>
</services>
<servers>
  <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool" gracefulStopAwaitMillis="30000"/>
  <server name="ams-monitor-server" services="ams-monitor-service" endpoints="ams-monitor" executors="http-thread-pool"/>
</servers>
<containers>
```

6.8 配置慢線程堆疊跟蹤

應用伺服器支援配置慢線程堆疊跟蹤。

\${APUSIC_HOME}/conf/apusic.conf 文件，在 endpoint 模組下將長線程監控開啟配置，如下：

```
enabledLongThread="true"
longThreadLogPath="/opt/AAMS-10.0.0/logs/longThread"
```

```
longThreadThreshold="60000" （長線程閾值，單位毫秒，默認值60000）  
longThreadRollingInterval="300000" （日誌滾動時間間隔，單位毫秒，默認值300000）  
longThreadCycleTime="3000" （長線程檢測週期，單位毫秒，默認值3000）
```

6.9 共享庫配置

共享庫可以在多個應用中共享相同的類庫，降低記憶體的使用。配置的步驟如下：

1. 可以在 `${APUSIC_HOME}/` 目錄下建立一個目錄，如 `sharelib`;
2. 目錄中放置需要在多個應用中共享的 jar 包；
3. 配置 `${APUSIC_HOME}/conf` 下的 `apusic.properties`，修改 `shared.loader` 的配置為：`shared.loader="${apusic.home}/sharelib/*.jar"`

經過上面的步驟即可完成共享庫配置，應用則可以直接使用該 jar 的資源，共享庫的類庫優先順序低於應用中相同的類庫優先順序。

7 安全配置

7.1 配置 HTTPS

應用伺服器支援配置HTTPS，支援單向認證和雙向認證，相容JKS、PEM格式證書。

注意：如果配置了新的 `endpoint` 需要在 `server` 中進行引用，配置了線程池也需要在 `server` 中進行引用。

以下示例中的證書文件路徑（如 `conf/cert/`）為示意路徑，需自行建立目錄並將證書文件放入，產品出廠默認證書為 `conf/keystore.jks`、`conf/cacerts.jks`。

7.1.1 單向認證 -- 證書格式為 JKS

應用伺服器支援配置格式為JKS的證書。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
  <SSLHostConfig>
    <Certificate certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
certificateKeystoreType="JKS"
      certificateKeystorePassword="[YOUR_PASSWORD]" />
    </SSLHostConfig>
  </endpoint>
```

表15：金蝶Apusic應用伺服器V10標準版證書格式為JKS的單向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
executor	使用的線程池	字元	
certificateKeystoreType	keystore 格式，JKS 或者 PKCS12	字元	
certificateKeystorePassword	keystore 密碼	字元	
certificateKeystoreFile	keystore 文件	字元	

7.1.2 雙向認證 -- 證書格式為 JKS

應用伺服器支援配置格式為JKS的證書，同時支援雙向認證。根據證書生成方式，雙向認證一般需要有對應的Ukey客戶端結合使用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
  <SSLHostConfig truststoreType="JKS" truststorePassword="[YOUR_PASSWORD]"
truststoreFile="conf/cert/arsp_server_truststore.jks" certificateVerification="required">
    <Certificate certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
certificateKeystoreType="JKS"
      certificateKeystorePassword="[YOUR_PASSWORD]" />
    </SSLHostConfig>
  </endpoint>
```

表16：金蝶Apusic應用伺服器V10標準版證書格式為JKS的雙向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
executor	使用的線程池	字元	
truststoreType	信任庫格式，JKS 或者 PKCS12	字元	
truststorePassword	信任庫密碼	字元	
truststoreFile	信任庫文件	字元	

certificateVerification	是否需要驗證客戶端證書，即是否雙向認證，默認不驗證	字元	
certificateKeystoreType	keystore 格式，JKS 或者 PKCS12	字元	
certificateKeystorePassword	keystore 密碼	字元	
certificateKeystoreFile	keystore 文件	字元	

7.1.3 單向認證 -- 證書格式為 PEM

應用伺服器支援配置格式為PEM的證書。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
  <SSLHostConfig>
    <Certificate certificateKeyFile="conf/cert/private_key.pem" certificateKeyPassword="[YOUR_PASSWORD]"
      certificateFile="conf/cert/public_key.pem" certificateChainFile="conf/cert/chain.pem"
      type="RSA" />
    </SSLHostConfig>
  </endpoint>
```

表17：金蝶Apusic應用伺服器V10標準版證書格式為PEM的單向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
executor	使用的線程池	字元	
type	證書型別	字元	
certificateKeyFile	私鑰文件	字元	
certificateKeyPassword	私鑰密碼，可選	字元	
certificateFile	公鑰文件	字元	
certificateChainFile	證書鏈文件，可選	字元	

7.1.4 雙向認證 -- 證書格式為 PEM

應用伺服器支援配置格式為PEM的證書，並支援雙向認證。根據證書生成方式，雙向認證一般需要有對應的Ukey客戶端結合使用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
  <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
  <SSLHostConfig caCertificateFile="conf/cert/public_ca_key.pem" certificateVerification="required">
    <Certificate certificateKeyFile="conf/cert/private_key.pem" certificateKeyPassword="[YOUR_PASSWORD]"
      certificateFile="conf/cert/public_key.pem" certificateChainFile="conf/cert/chain.pem"
      type="RSA" />
    </SSLHostConfig>
  </endpoint>
```

表18：金蝶Apusic應用伺服器V10標準版證書格式為PEM的雙向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
executor	使用的線程池	字元	
type	證書型別	字元	
certificateKeyFile	私鑰文件	字元	

certificateKeyPassword	私鑰密碼，可選	字元	
certificateFile	公鑰文件	字元	
certificateChainFile	證書鏈文件，可選	字元	
caCertificateFile	根證書文件，可選	字元	
certificateVerification	是否需要驗證客戶端證書，即是否雙向認證	字元	

7.2 國密配置說明

7.2.1 環境說明

需要使用 JDK 8 版本和國密瀏覽器（如 360 國密瀏覽器）。

需要準備國密證書。

7.2.2 應用伺服器證書配置

7.2.2.1 單向認證

修改 `apusic.conf` 文件，增加如下的 `endpoint` 配置（如果已經存在，則進行修改）：

```
<endpoint name="https" port="6887" maxThreads="150" SSLEnabled="true">
  <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1">
    <Certificate certificateKeystoreFile="conf/keystore.pfx" certificateKeystorePassword="[YOUR_PASSWORD]"
      certificateKeyPassword="[YOUR_PASSWORD]" type="EC" certificateKeystoreType="PKCS12" />
  </SSLHostConfig>
</endpoint>
```

表19：金蝶Apusic應用伺服器V10標準版國密證書的單向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
certificateKeystoreFile	表示證書文件的路徑	字元	
protocols	需要指定為 SMv1.1	字元	
sslProtocol	需要指定為 SMv1.1	字元	
certificateKeystorePassword	為證書庫文件的密碼	字元	
certificateKeyPassword	為證書的密碼	字元	
type	證書型別	字元	
certificateKeystoreType	keystore型別，如PKCS12	字元	

7.2.2.2 雙向認證

修改 `apusic.conf` 文件，增加如下的 `endpoint` 配置（如果已經存在，則進行修改）：

```
<endpoint name="https" port="6887" maxThreads="150" SSLEnabled="true">
  <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1" truststoreFile="conf/truststore.jks"
    truststorePassword="[YOUR_PASSWORD]" certificateVerification="want" truststoreType="JKS">
    <Certificate certificateKeystoreFile="conf/keystore.pl2" certificateKeystorePassword="[YOUR_PASSWORD]"
      certificateKeyPassword="[YOUR_PASSWORD]" type="EC" certificateKeystoreType="PKCS12" />
  </SSLHostConfig>
</endpoint>
```

表20：金蝶Apusic應用伺服器V10標準版國密證書的雙向認證方式配置說明

配置引數	說明	值型別	默認值
SSLEnabled	是否是 ssl 通訊	布林值	
certificateKeystoreFile	表示證書文件的路徑	字元	
protocols	需要指定為 SMv1.1	字元	
sslProtocol	需要指定為 SMv1.1	字元	
certificateKeystorePassword	為證書庫文件的密碼	字元	
certificateKeyPassword	為證書的密碼	字元	
type	證書型別	字元	
certificateKeystoreType	keystore型別，如PKCS12	字元	
truststoreFile	指定信任庫文件，其型別是JKS	字元	
truststorePassword	信任庫密碼	字元	
certificateVerification	是否進行證書驗證，雙向認證時設置為 required，可選客戶端認證設置為 optional 或 want	字元	

7.2.2.3 增加 endpoint 到 server

修改 `apusic.conf` 文件，修改如下的配置，其中 `endpoints` 屬性中的 `https` 就是上面配置單向認證或雙向認證的 `endpoint` 的名字：

```
<server name="ams-server" services="ams-service" endpoints="ams-http,https" executors="http-thread-pool" />
```

7.2.3 啟動應用伺服器

通過上面的步驟配置完成後，通過命令 `apusic.sh gm start` 或 `apusic.sh gm run`（Windows 下為 `apusic.bat gm start` 或 `apusic.bat gm run`）可以啟動國密支援的容器。

備註：可以通過在啟動指令碼中增加引數 `-Djavax.net.debug=ssl:handshake` 來列印具體的 SSL 處理過程，方便分析可能存在的問題。

7.2.4 客戶端配置

專案配置國密SSL協議之後，需要使用支援國密SSL協議的瀏覽器訪問，例如360安全瀏覽器、奇安信等。瀏覽器需要開啟國密通訊功能。

然後需要在客戶端中匯入根證書，例如 `sm2.ca.pem`。

以360安全瀏覽器為例，進入“設置”-“選項”-“安全設置”，勾選“國密通訊協議”，設置“國密SSL通訊白名單”。

“隱私安全設置”-“管理HTTPS/SSL證書...”，“受信任的根證書頒發機構”，匯入根證書。

重啟瀏覽器。

7.2.5 其他問題

如果使用 Oracle JDK，由於 JVM 默認的加密強度限制，需要下載 JCE 無限強度加密策略文件（Unlimited Strength Jurisdiction Policy Files），解壓後用其中的 `local_policy.jar` 和 `US_export_policy.jar` 替換 `%JAVA_HOME%/jre/lib/security/` 目錄下的同名文件，解除加密強度限制。

[JCE 文件](#)

7.3 慢攻擊防護

該功能是防禦慢速HTTP連線攻擊，針對http慢讀、慢寫等異常行為進行攔截或者告警；

疑似慢攻擊判定條件：

- (1) 請求行 + 請求頭 超過指定時間（超時會關閉連線，記錄事件）
- (2) 包傳送間隔超過指定時間（超時不會關閉連線，記錄事件）

慢攻擊支援配置告警模式或保護模式：

(1) 告警模式：

- a、檢測到疑似慢攻擊，如果超過指定次數將加入黑名單並列印日誌；不超過指定次數則不列印日誌，不加入黑名單
- b、黑名單中的慢攻擊過期時間，期間沒有發生慢攻擊則移出黑名單，並列印相關日誌；如果發生慢攻擊則更新黑名單過期時間（不管是否超過閾值，當前ip有新請求時會即時判斷是否過期，如果沒有請求則後臺線程非同步檢測，每隔30秒檢測一次）
- c、發起新的請求時，如果該IP在黑名單時，不進行攔截（不關閉連線）

(2) 防護模式：

- a、檢測到疑似慢攻擊，如果超過指定次數將加入黑名單並列印日誌；不超過指定次數則不列印日誌，不加入黑名單
- b、黑名單中的慢攻擊過期時，列印相關日誌（當前ip有新請求時會即時判斷是否過期，如果沒有請求則後臺線程非同步檢測，每隔30秒檢測一次）
- c、發起新的請求時，如果該IP在黑名單時，進行攔截（關閉連線）

慢攻擊常用配置引數：

- (1) 疑似慢攻擊閾值：超過閾值則加入黑名單
- (2) 黑名單過期時間：ip加入黑名單後，超過該時間將從黑名單中移除
- (3) 慢攻擊時間視窗：指定時間視窗內發生次數超過閾值則加入黑名單
- (4) 包傳送間隔超時時間
- (5) 請求行 + 請求頭超時時間
- (6) 慢攻擊模式：告警模式/保護模式

- `connectionTimeout` 讀取整個請求行 + 請求頭的超時時間（未讀取完則會記錄疑似慢攻擊事件，關閉連線）
- `sdosBlacklistEnable="false"` 是否開啟黑名單
- `sdosThreshold="3"` 疑似慢攻擊閾值
- `sdosTimeWindowMinutes="30"` 指定時間視窗內發生疑似慢攻擊次數大於閾值則加入黑名單
- `sdosBlacklistTtlMinutes="30"` 黑名單過期時間（出廠默認30分鐘），（後面非同步檢測和重新整理，輸出日誌的時間會有誤差，如果一直有請求則發現過期就會輸出過期日誌）
- `sdosMode="0"` 防護模式，小於等於0 告警模式；大於0 防護模式
- `sdosMaxPacketInterval="10000"` 最大包傳送間隔，兩個包之間間隔超過會被記錄為攻擊事件（毫秒，小於等於0不起效；出廠默認10000）
- `sdosBlacklistFlushMinutes="10"` 多長時間重新整理黑名單到文件中，0 表示即時重新整理

注意：

- (1) 後臺清理過期事件和黑名單是非同步進行的，後臺定時器每隔30秒執行一次
- (2) 如果當前請求ip在黑名單中，會判斷是否過期，如果過期會立即清理
- (3) 記錄疑似慢攻擊事件時，當前請求ip在黑名單中，則不會記錄事件，不在黑名單才會記錄事件，超過閾值則加入黑名單，清空記錄事件
- (4) 黑名單文件默認非即時重新整理，默認10分鐘重新整理一次，`sdosBlacklistFlushMinutes` 配置為0才即時重新整理
- (5) 黑名單隻針對新連線生效
- (6) 包最大間隔時間觸發不會關閉連線

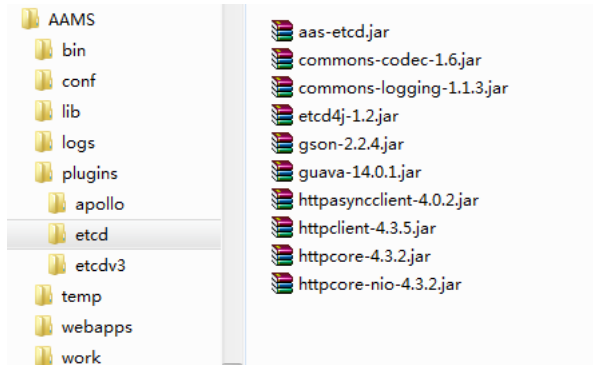
```

<endpoints>
  <endpoint name="ams-http"
    port="6888"
    protocol="HTTP/1.1"
    connectionTimeout="30000"
    redirectPort="6443"
    executor="http-thread-pool"
    maxConnections="10000"
    enableLookups="false"
    acceptCount="100"
    maxPostSize="104857600"
    maxHttpRequestSize="8192"
    compression="on"
    disableUploadTimeout="true"
    compressionMinSize="2048"
    compressibleMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/xml,application/atom+xml,application/mpeg,video/mp4"
    URIEncoding="utf-8"
    processorCache="20000"
    tcpNoDelay="true"
    keepAliveTimeout="15000"
    maxKeepAliveRequests="100"
    longThreadThreshold="60000"
    longThreadCycleTime="3000"
    longThreadRollingInterval="300000"
    enabledLongThread="false"
    sdosBlacklistEnable="true"
    sdosThreshold="3"
    sdosTimeWindowMinutes="30"
    sdosBlacklistTtlMinutes="3"
    sdosMode="1"
    sdosMaxPacketInterval="5000"
    sdosBlacklistFlushMinutes="0"
  >
    </endpoint>
  <endpoint name="ams-jmx"
    port="6868"
    auth="false"
    protocol="JMXRMI/2.0">
</endpoints>

```

8 集中配置

應用伺服器支援集中配置軟體包括 ETCD 和 apollo，後續會增加更多的支援。對集中配置軟體的實現包，放在產品包的 `plugins` 目錄下，如下面圖所示：



備註：一般情況下一個用戶場景只會使用到一個實現，可以刪除其他的實現，以節約空間。

8.1 使用 apollo 集中配置

下面介紹如何在金蝶Apusic應用伺服器標準版中使用 apollo 軟體進行集中配置。

8.1.1 新增集中配置服務

在 `${APUSIC_HOME}/conf/apusic.conf` 配置文件增加 `plugins` 的配置，並指定 apollo 的實現，示例IP `1.x.x.231` 如下圖所示：

```
<plugins>
  <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test"
    metaUrl="http://1.x.x.231:8080" namespace="test-namespace" />
</plugins>
```

```

1 1 <?xml version="1.0" encoding="utf-8" standalone="no"?><config>
2
3 2 <plugins>
4   <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test"
      metaUrl="http://172.20.129.231:8080" namespace="test-namespaces"/>
5 </plugins>
6
7 3 <listeners>
8   <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
9   <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
10  <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
11  <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
12  <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
13 </listeners>
14
15 4 <resources>
16   <resource auth="Container" description="User database that can be updated and saved"
      factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase" pathname="conf/aas-users.xml"
      type="com.apusic.ams.UserDatabase"/>
17 </resources>
18
19 5 <executors>
20   <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-"
      prestartminSpareThreads="true"/>
21 </executors>
22
23 6 <endpoints>
24   <endpoint URIEncoding="utf-8" acceptCount="100"
      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image/png,image/gif,video/mp4" compression="on"
      compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192"
      maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http" port="6999" processorCache="20000" protocol="HTTP/1.1"
      redirectPort="7443" tcpNoDelay="true">
25 </endpoint>
26 </endpoints>
27
28 7 <services>
29   <service defaultHost="localhost" name="ams-service">
30     <realm className="com.apusic.ams.realm.LockOutRealm">
31       <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
32     </realm>
33     <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
34   </host>
35 </service>
36

```

其中 `className="com.apusic.ams.config.listener.ApolloPluginLoader"` 是載入 apollo 配置的類，屬性包括了：

表21：金蝶Apusic應用伺服器V10標準版與Apollo整合配置說明

配置引數	說明	值型別	默認值
appId	配置的應用id	字元	
cluster	配置屬於的叢集名稱	字元	
metaUrl	Apollo 提供服務的地址	字元	
namespace	配置的名稱空間	字元	
env	配置的環境（開發環境，正式環境等等）	字元	
idc	配置的資料中心	字元	

以上屬性可通過配置文件配置，也可通過環境變數配置，也可通過系統變數設置。

優先順序是：環境變數 > 系統變數 > 配置文件

一般環境變數和系統變數的變數名稱為大寫的，環境變數（`System.getProperty("xxx")` 獲取）和系統變數（`System.getenv("xxx")` 獲取）的變數名為如下，每個變數名會與上面的屬性對應：

表22：金蝶Apusic應用伺服器V10標準版與Apollo整合環境變數配置說明

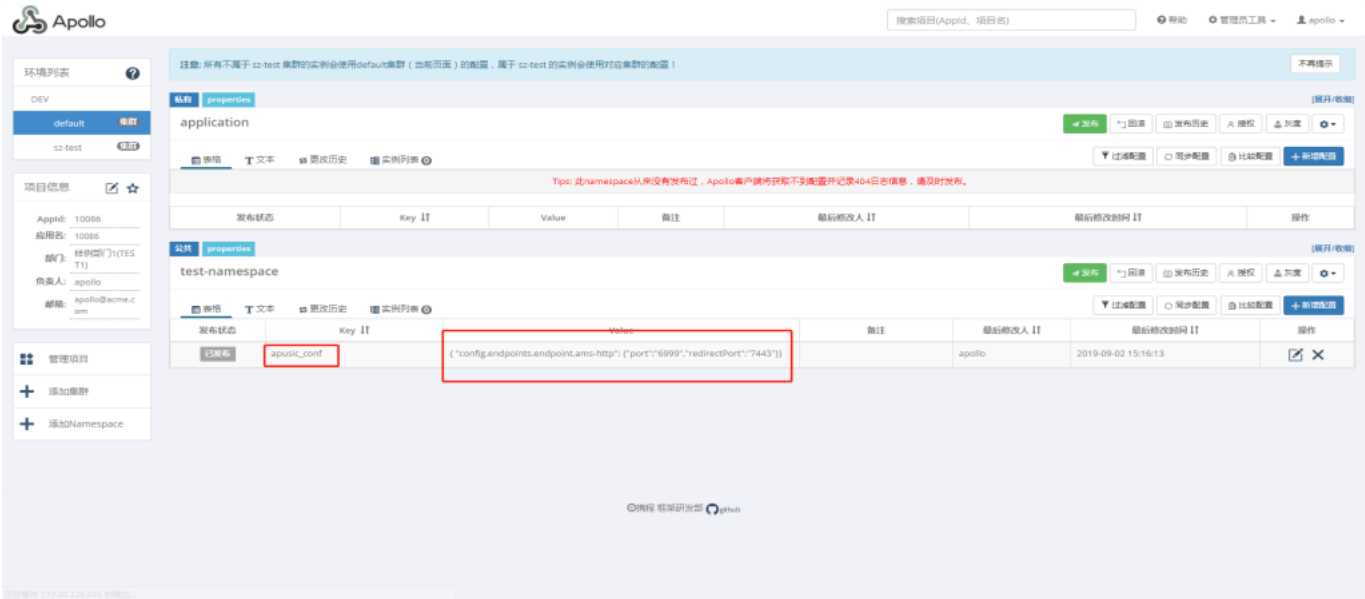
配置引數	說明	值型別	默認值
AAS_CONFIG_APOLLO_APP_ID	配置的應用id	字元	
AAS_CONFIG_APOLLO_CLUSTER	配置屬於的叢集名稱	字元	
AAS_CONFIG_APOLLO_META_SERVER	Apollo 提供服務的地址	字元	

AAS_CONFIG_APOLLO_NAMESPACE	配置的名稱空間	字元	
AAS_CONFIG_APOLLO_ENV	配置的環境（開發環境，正式環境等等）	字元	
AAS_CONFIG_APOLLO_IDC	配置的資料中心	字元	

8.1.2 在 apollo 的配置

8.1.2.1 apusic.conf 配置

與配置中心整合後，應用伺服器的配置文件（`apusic.conf` 文件）可通過設置 `key` 為 `apusic_config`，`value` 為要改變的值進行修改，如下圖：



其中 `key` 必須為 `apusic_config`，`value` 必須是 json 字串。

```
{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}
```

`config.endpoints.endpoint.ams-http` 拆分對應這 `config > endpoints > endpoint > ams-http` 對應這 xml 的層級結構：

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6   </plugins>
7
8   <listeners>
9     <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
10    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
11    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
12    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
13    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
14  </listeners>
15
16  <resources>
17    <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
18      pathName="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
19  </resources>
20
21  <executors>
22    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
23  </executors>
24
25  <endpoints>
26    <endpoint JRIEncoding="utf-8" acceptCount="100"
27      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
28      /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
29      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
30      port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
31    </endpoint>
32  </endpoints>
33
34  <services>
35    <service defaultHost="localhost" name="ams-service">
36      <realm className="com.apusic.ams.realm.LockOutRealm">
37        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
38      </realm>
39      <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
40      </host>
41    </service>
42  </services>
43
44  <servers>
45    <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
46  </servers>
47
48  <containers>
49    <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
50      name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
51  </containers>
52 </config>

```

port 和 redirectPort 對應著屬性。

```

{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}

```

上圖的意思就是，修改名稱為 ams-http 的 xml 標籤為 endpoint 的 port 屬性的值為 6999，redirectPort 的屬性值為 7443。

8.1.2.2 數據源的配置

通過 apollo 軟體的操作介面進行配置，資料的格式可以參考 ETCD 章節下的說明。

8.1.2.3 日誌的配置

通過 apollo 軟體的操作介面進行配置，資料的格式可以參考 ETCD 章節下的說明。

8.1.2.4 JVM 的配置

通過 apollo 軟體的操作介面進行配置，資料的格式可以參考 ETCD 章節下的說明。

8.2 使用 ETCD 集中配置

下面介紹如何在金蝶Apusic應用伺服器標準版中使用ETCD進行集中配置。

8.2.1 新增集中配置服務

在 \${APUSIC_HOME}/conf/apusic.conf 配置文件增加 plugins 的配置，並指定 etcd 的配置實現，示例IP 172.x.x.74。

要支援 ETCD2，則需要指定實現為：

```
<plugins>
  <plugin className="com.apusic.ams.config.listener.EtcdPluginLoader" clusterId="cluster"
host="172.x.x.74"
    port="2379" namespace="test" />
</plugins>
```

要支援 ETCD3 則需要指定為：

```
<plugins>
  <plugin className="com.apusic.ams.config.listener.EtcdV3PluginLoader" clusterId="cluster"
host="172.x.x.74"
    port="2379" namespace="test" />
</plugins>
```

示例如下圖：



```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6
7     <plugin className="com.apusic.ams.config.listener.EtcdPluginLoader" host="xxx" port="xxx" namespace="test-namespace" clusterId="xxx"/>
8   </plugins>
9
10  <listeners>
11    <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
12    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
13    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
14    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
15    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
16  </listeners>
17
18  <resources>
19    <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
20      pathname="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
21  </resources>
22
23  <executors>
24    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
25  </executors>
26
27  <endpoints>
28    <endpoint URIEncoding="utf-8" acceptCount="100"
29      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
30      /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
31      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
32      port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true"/>
33  </endpoint>
34  </endpoints>
35
36  <services>
37    <service defaultHost="localhost" name="ams-service">
38      <realm className="com.apusic.ams.realm.LockOutRealm">
39        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
40      </realm>
41      <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
42      </host>
43    </service>
44  </services>
45
46  <servers>
47    <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
48  </servers>
49
50  <containers>
51    <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
52      name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
53  </containers>
54 </config>
```

其中 `className="com.apusic.ams.config.listener.EtcdPluginLoader"` 是載入 etcd 配置的類，屬性包括了屬性說明：

表23：金蝶Apusic應用伺服器V10標準版與ETCD整合配置說明

配置引數	說明	值型別	默認值
------	----	-----	-----

host	安裝 etcd 的伺服器 IP 地址, 必須	字元	
port	etcd 服務的客戶端訪問端口, 必須	字元	
namespace	應用的名稱空間, 必須	字元	
clusterId	應用的叢集標識, 必須	字元	

以上屬性可通過配置文件配置, 也可通過環境變數配置, 也可通過系統變數設置

優先順序是: 環境變數 > 系統變數 > 配置文件

一般環境變數和系統變數的變數名稱為大寫的, 環境變數 (`System.getProperty("xxx")` 獲取) 和系統變數 (`System.getenv("xxx")` 獲取) 的變數名為如下, 每個變數名會與上面的屬性對應:

表24: 金蝶Apusic應用伺服器V10標準版與ETCD整合環境變數配置說明

配置引數	說明	值型別	默認值
AAS_CONFIG_ETCD_HOST	安裝 etcd 的伺服器 IP 地址, 必須	字元	
AAS_CONFIG_ETCD_PORT	etcd 服務的客戶端訪問端口, 必須	字元	
AAS_CONFIG_ETCD_NAMESPACE	應用的名稱空間, 必須	字元	
AAS_CONFIG_ETCD_CLUSTERID	應用的叢集標識, 必須	字元	

8.2.2 在 ETCD 的配置

8.2.2.1 apusic.conf 的配置

與配置中心整合後, 應用伺服器的配置文件 (`apusic.conf` 文件) 可通過設置 `key` 為 `apusic_config`, `value` 為要改變的值, 進行修改。

在 ETCD 伺服器中設置如下的值, 則表示修改端口為 6999 和 7443:

```
./etcdctl put /test/cluster/aamsconfig/apusic_config '{"config.endpoints.endpoint.ams-http": {"port": "6999", "redirectPort": "7443"} }'
```

配置資訊最後儲存在 `conf` 目錄下的 `apusic.conf` 文件。

`config.endpoints.endpoint.ams-http` 拆分對應這 `config > endpoints > endpoint > ams-http` 對應這 xml 的層級結構:

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6   </plugins>
7
8   <listeners>
9     <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
10    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
11    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
12    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
13    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
14  </listeners>
15
16  <resources>
17    <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
18      pathName="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
19  </resources>
20
21  <executors>
22    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
23  </executors>
24
25  <endpoints>
26    <endpoint JRIEncoding="utf-8" acceptCount="100"
27      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
28      /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
29      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
30      port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
31    </endpoint>
32  </endpoints>
33
34  <services>
35    <service defaultHost="localhost" name="ams-service">
36      <realm className="com.apusic.ams.realm.LockOutRealm">
37        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
38      </realm>
39      <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
40      </host>
41    </service>
42  </services>
43
44  <servers>
45    <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
46  </servers>
47
48  <containers>
49    <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
50      name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
51  </containers>
52 </config>

```

port 和 redirectPort 對應著屬性。

```

{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}

```

上圖的意思就是，修改名為 ams-http 的 xml 標籤為 endpoint 的 port 屬性的值為 6999，redirectPort 的屬性值為 7443。

8.2.2.2 數據源的配置

與配置中心整合後，數據源的配置為 datasource_config，它是一個 json 格式的字串。可以在 etcd 中增加如下的配置：

```

./etcdctl put /test/cluster/aamsconfig/datasource_config '{"jdbc/test":
{"auth":"Container","type":"javax.sql.DataSource","maxActive":"100","username":"root","password":"
[YOUR_PASSWORD]","driverClassName":"com.mysql.cj.jdbc.Driver","url":"jdbc:mysql://127.0.0.1:3306/test","max

```

其中 jdbc/test 為數據源的名稱，如果存在該名稱，則進行修改動作，如果不存在該名稱，則進行新增的動作。配置資訊最後儲存在 conf 目錄下的 context.xml 文件。

8.2.2.3 日誌的配置

與配置中心整合後，日誌的配置為 logging_conf，它是一個 json 格式的字串。可以在 etcd 中增加如下類似的配置，修改多個配置，則增加多個 json 格式的 key-value：

```
./etcdctl put /test/cluster/aamsconfig/logging_conf '{"com.apusic.connector.http2.level":"FINER"}'
```

配置資訊最後儲存在 `conf` 目錄下的 `logging.properties` 文件。

8.2.2.4 JVM 的配置

與配置中心整合後，日誌的配置的 `key` 為 `jvm_opt`，它是一個普通格式的字串，不是一個 json 字串，配置的格式類似如下：

```
./etcdctl put /test/cluster/aamsconfig/jvm_opt '-server -Djava.net.preferIPv4Stack=true -Xms513m -Xmx1023m -XX:MaxPermSize=257m'
```

配置資訊最後儲存在 `bin` 目錄下的 `apusic.bat` (window 下) 或 `apusic.sh` (linux 下) 文件。

9 Session 儲存與叢集

9.1 配置 Session 儲存

金蝶Apusic應用伺服器標準版單機方式有兩種 Session 儲存方式：`StandardManager`（默認）、`PersistentManager`。

- `StandardManager` 方式: 把 Session 放在記憶體中，停止時才把記憶體中的 Session 儲存到文件，再次啟動時從文件讀取。
- `PersistentManager` 方式: 又分為文件或者資料庫儲存。

9.1.1 配置本地記憶體儲存

默認的 Session 儲存就是本地記憶體儲存（`StandardManager`），該方式 Session 是放在記憶體中，停止時才把記憶體中的 Session 儲存到文件，再次啟動時從文件讀取。

配置如下：

```
<Manager maxActiveSessions="-1">
</Manager>

<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true">
        <Manager
          maxActiveSession="-1">
        </Manager>
      </application>
    </host>
  </service>
</services>
```

配置說明：

表25：金蝶Apusic應用伺服器V10標準版配置本地記憶體儲存引數說明

配置引數	說明	值型別	默認值
maxActiveSessions	最大會話數，-1 表示不限制，超過後再建立 Session 會直接拋異常	整型	-1

9.1.2 配置本地文件儲存

使用 `PersistentManager` 方式配置本地文件儲存，如下：

```
<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true">
        <Manager className="com.apusic.ams.session.PersistentManager"
          saveOnRestart="true"
          maxActiveSessions="-1"
          minIdleSwap="-1"
          maxIdleSwap="-1"
          maxIdleBackup="60">
          <Store className="com.apusic.ams.session.FileStore"/>
        </Manager>
      </application>
    </host>
  </service>
</services>
```

配置說明：

表26：金蝶Apusic應用伺服器V10標準版配置本地文件儲存引數說明

配置引數	說明	值型別	默認值
saveOnRestart	配置服務重啟對 Session 的處理，若為 true，則關閉前把有效的 Session 儲存，啟動後重新載入	布林值	

maxActiveSessions	活動狀態 Session 的最大數，為 -1 時則不限制，否則 Session Manager 將會把超出的 Session 對象轉移到 Session Store 中	整型	
minIdleSwap	Session 不活動的最短時間，超過該時間，Session Manager 可能會把該 Session 對象轉移到 Session Store 中，單位為秒，該 Session 將不在記憶體中，如果配置了 minIdleSwap 則兩個條件都需要滿足	整型	
maxIdleSwap	Session 不活動的最長時間，超過該時間，Session Manager 將會把該 Session 對象轉移到 Session Store 中，該 Session 將不在記憶體中。如果配置了 minIdleSwap 則兩個條件都需要滿足	整型	
maxIdleBackup	Session 不活動的最長時間，超過該時間，Session Manager 將會把該 Session 對象備份到 Session Store 中，但該 Session 對象依然存在記憶體中	整型	

Store 標籤配置如下：

表27：金蝶Apusic應用伺服器V10標準版配置本地文件儲存store標籤配置引數說明

配置引數	說明	值型別	默認值
className	儲存方式類名	字元	com.apusic.ams.session.FileStore

9.1.3 配置資料庫儲存

使用 PersistentManager 方式配置資料庫儲存，需要事先在資料庫建立表，如下：

```
<Manager className="com.apusic.ams.session.PersistentManager" saveOnRestart="true" maxActiveSessions="1"
minIdleSwap="-1" maxIdleSwap="-1" maxIdleBackup="60">
  <Store className="com.apusic.ams.session.JDBCStore" driverName="com.mysql.cj.jdbc.Driver"
connectionURL="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&characterEncoding=utf-8"
connectionName="myuser" connectionPassword="[YOUR_PASSWORD]" sessionTable="ams_session"
sessionIdCol="session_id"
sessionDataCol="session_data" sessionValidCol="session_valid"
sessionMaxInactiveCol="max_inactive"
sessionLastAccessedCol="last_access" sessionAppCol="app_name" />
</Manager>
```

表28：金蝶Apusic應用伺服器V10標準版配置資料庫儲存引數說明

配置引數	說明	值型別	默認值
saveOnRestart	配置服務重啟對 Session 的處理，若為 true，則關閉前把有效的 Session 儲存，啟動後重新載入	布林值	
maxActiveSessions	活動狀態 Session 的最大數，為 -1 時則不限制，否則 Session Manager 將會把超出的 Session 對象轉移到 Session Store 中	整型	
minIdleSwap	Session 不活動的最短時間，超過該時間，Session Manager 可能會把該 Session 對象轉移到 Session Store 中，單位為秒，該 Session 將不在記憶體中，如果配置了 minIdleSwap 則兩個條件都需要滿足	整型	
maxIdleSwap	Session 不活動的最長時間，超過該時間，Session Manager 將會把該 Session 對象轉移到 Session Store 中，該 Session 將不在記憶體中。如果配置了 minIdleSwap 則兩個條件都需要滿足	整型	
maxIdleBackup	Session 不活動的最長時間，超過該時間，Session Manager 將會把該 Session 對象備份到 Session Store 中，但該 Session 對象依然存在記憶體中	整型	

Store標籤中配置如下：

表29：金蝶Apusic應用伺服器V10標準版配置資料庫文件儲存store標籤引數說明

配置引數	說明	值型別	默認值
------	----	-----	-----

className	儲存方式類名	字元	com.apusic.ams.session.JDBCStore
driverName	資料庫驅動類名，根據數據源型別填寫	字元	
connectionURL	連線數據源URL，根據數據源連線方式填寫	字元	
connectionName	連線數據源用戶名稱，需要有遠端連線許可權	字元	
connectionPassword	連線數據源用戶名稱密碼	字元	
sessionTable	session儲存表名，需要在資料庫中提前建立	字元	
sessionIdCol	session儲存ID列名，需要在資料庫中提前建立	字元	
sessionDataCol	session儲存資料列名，需要在資料庫中提前建立	字元	
sessionValidCol	session儲存值列名，需要在資料庫中提前建立	字元	
sessionMaxInactiveCol	非活動最大值列名，需要在資料庫中提前建立	字元	
sessionLastAccessedCol	最後儲存資料列名，需要在資料庫中提前建立	字元	
sessionAppCol	應用名稱列名，需要在資料庫中提前建立	字元	

9.2 叢集方案

9.2.1 Session 組播同步

Session 同步需要滿足以下幾個要求：

- 1.Session 的屬性必須實現 `java.io.Serializable` 介面
- 2.負載均衡器需要配置會話粘滯模式（sticky Session mode），測試時可不開啟會話粘滯
- 3.多臺機器時間需要同步
- 4.保證多播端口可用
- 5.應用的 `web.xml` 需要加入 `<distributable/>`。
- 6.應用使用目錄方式部署

9.2.1.1 新增叢集配置

在需要加入叢集的機器中的 `apusic.conf` 文件的 `host` 或者 `service` 裡面新增如下內容（注意：在 `service` 下新增表示對所有虛擬主機均啟用叢集功能，否則對某個虛擬主機啟用叢集功能）。示例IP 224.x.x.4、172.x.x.1。

```
<Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster" channelSendOptions="6">
  <Manager className="com.apusic.ams.ha.session.BackupManager" expireSessionsOnShutdown="false"
    notifyListenersOnReplication="true" />
  <Channel className="com.apusic.ams.tribes.group.GroupChannel">
    <Membership className="com.apusic.ams.tribes.membership.McastService" address="224.x.x.4"
port="45564"
    frequency="500" dropTime="3000" />
    <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver" address="172.x.x.1"
port="5000"
    selectorTimeout="100" maxThreads="6" />
    <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
      <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender" />
    </Sender>
    <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector" />
    <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatchInterceptor" />
    <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor" />
  </Channel>
  <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter="" />
  <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve" />
</Cluster>
```

```
<ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener" />
</Cluster>
```

```
<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext"/>
    </host>
    <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
      channelSendOptions="6">
      <Manager className="com.apusic.ams.ha.session.DeltaManager"
        expireSessionsOnShutdown="false"
        notifyListenersOnReplication="true"/>
      <Channel className="com.apusic.ams.tribes.group.GroupChannel">
        <Membership className="com.apusic.ams.tribes.membership.McastService"
          address="228.0.0.4"
          port="45564"
          frequency="500"
          dropTime="3000"/>
        <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
          address="auto"
          port="5000"
          selectorTimeout="100"
          maxThreads="6"/>
        <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
          <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
        </Sender>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
      </Channel>
      <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
      <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
      <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
    </Cluster>
  </service>
</services>

<servers>
  <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
</servers>
```

注意: 需要把 NioReceiver 裡面的 address 修改為當前機器的 IP 地址。

重要配置詳解:

Cluster: 應用伺服器叢集配置。

表30: 金蝶Apusic應用伺服器V10標準版叢集配置cluster標籤說明

配置引數	說明	值型別	默認值
className	實現叢集功能的類名稱	字元	com.apusic.ams.ha.tcp.SimpleTcpCluster
channelSendOptions	訊息傳送選項，可以設置為 2（確認傳送），4（同步傳送），8（非同步傳送）以及它們的邏輯或運算結果，默認是 8，如果希望非同步傳送且確認傳送結果，那麼設置為 2+8=10	字元	

Manager: 叢集會話管理器。

表31: 金蝶Apusic應用伺服器V10標準版叢集配置manager標籤說明

配置引數	說明	值型別	默認值
className	叢集會話管理器類，會話改變了會同步給叢集中的其他節點	字元	com.apusic.ams.ha.session.BackupManager
expireSessionsOnShutdown	設置為 true 時，一個節點關閉，將導致叢集下的所有 Session 失效	布林值	false
notifyListenersOnReplication	如果希望當叢集中的節點上覆制或刪除會話屬性時通知會話監聽器就設置 notifyListenersOnReplication 為 true	布林值	
mapSendOptions	為 6 表示 BackupManager 同步傳送訊息，參考 "Cluster"的 channelSendOptions	字元	

Channel: 是主元件，`channel` 管理一組子元件，並和它們一起組成了AAS例項間的通訊框架。在AAS叢集中，`DeltaManager` 通過`SimpleTcpCluster` 調用`channel`來實現資訊傳遞，而`BackupManager`自己調用`channel`以及子元件這些元件來實現資訊傳遞。`ReplicatedContext`也會調用`channel`傳遞`context`屬性。

表32：金蝶Apusic應用伺服器V10標準版叢集配置channel標籤說明

配置引數	說明	值型別	默認值
className	Channel類名	字元	com.apusic.ams.tribes.group.GroupChannel

****Membership:**** 叢集節點配置。

表33：金蝶Apusic應用伺服器V10標準版叢集配置membership標籤說明

配置引數	說明	值型別	默認值
className	叢集節點資訊管理類名	字元	com.apusic.ams.tribes.membership.McastService
address	多播地址	字元	
port	多播端口	字元	
frequency	傳送心跳時間間隔，單位毫秒	整型	
dropTime	該 dropTime 配置的時間內沒有收到某個心跳，那麼這個心跳所屬節點就會從當前節點維護的節點列表刪除，單位毫秒	整型	

****Receiver:**** 接收器，負責接收訊息。

表34：金蝶Apusic應用伺服器V10標準版叢集配置receiver標籤說明

配置引數	說明	值型別	默認值
className	訊息接收類名	字元	com.apusic.ams.tribes.transport.nio.NioReceiver
address	節點監聽地址	字元	
port	節點監聽端口	字元	
selectorTimeout	接收器的 selector 超時時間	整型	
maxThreads	最大線程數	整型	

****Sender:**** 傳送器，負責將訊息傳送給其他節點的 `Receiver`。

表35：金蝶Apusic應用伺服器V10標準版叢集配置sender標籤說明

配置引數	說明	值型別	默認值
className	Sender類名	字元	com.apusic.ams.tribes.transport.ReplicationTransmitter

****Interceptor:**** 攔截器。

表36：金蝶Apusic應用伺服器V10標準版叢集配置interceptor標籤說明

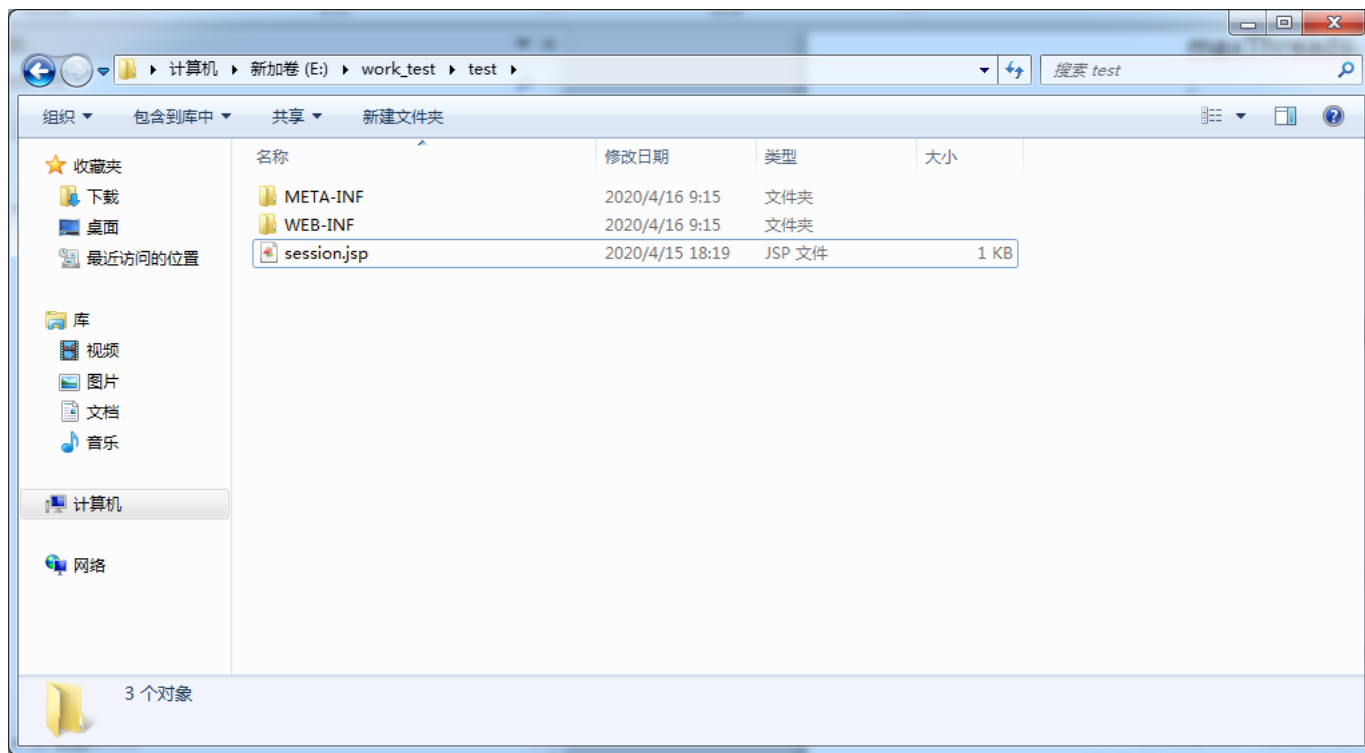
配置引數	說明	值型別	默認值
className	Interceptor類名	字元	

9.2.1.2 部署應用和設置 jvmRoute

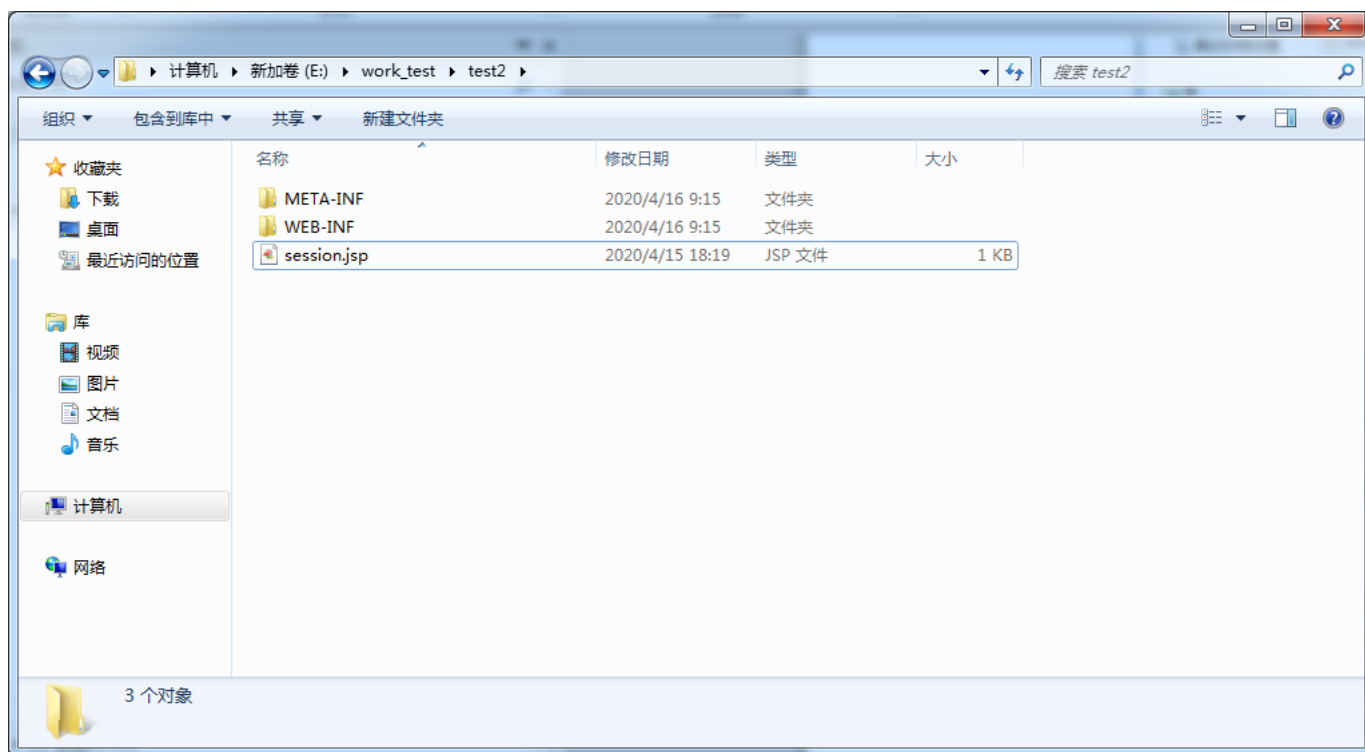
1.測試應用

test.war 是用於測試叢集的應用。裡面只有一個 session.jsp，需要把應用解壓進行部署。

節點 1 是解壓到 e:/work_test/test



節點 2 是解壓到 e:/work_test/test2



session.jsp 内容:

```

1 <%@ page import="java.text.SimpleDateFormat" %>
2 <%@ page import="java.util.Date" %>
3 <%@ page import="java.util.logging.Logger" %>
4 <%@ page language="java" contentType="text/html; charset=UTF-8"
5     pageEncoding="UTF-8"%>
6 <%
7     Logger logger = Logger.getLogger("test");
8     String str = (String) request.getSession().getAttribute("test");
9     if(null == str) {
10         str = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
11         request.getSession().setAttribute("test", str);
12         logger.info("session put: " + str);
13     } else {
14         logger.info("session get: " + str);
15     }
16 %>

```

從Session 裡面獲取資料，沒有資料再往裡放資料。

修改 web.xml 文件，新增 <distributable/> ;

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance http://www.w3.org/2001/XMLSchema-instance"
4     <welcome-file-list>
5         <welcome-file>index.html</welcome-file>
6         <welcome-file>index.htm</welcome-file>
7         <welcome-file>index.jsp</welcome-file>
8         <welcome-file>default.html</welcome-file>
9         <welcome-file>default.htm</welcome-file>
10        <welcome-file>default.jsp</welcome-file>
11    </welcome-file-list>
12
13    <distributable/>
14
15 </web-app>

```

2.部署應用和修改配置

修改 apusic.conf 文件的 service 加上 jvmRoute="nodeXXX" .

確保每個 jvmRoute 都是不同。還需要在 application 加上 className="com.apusic.ams.ha.context.ReplicatedContext" 。

這裡只有兩個節點，分別為 node01、node02

```

85
86 <services>
87   <service name="ams-service" defaultHost="localhost" jvmRoute="node02">
88     <realm className="com.apusic.ams.realm.LockOutRealm">
89       <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
90     </realm>
91     <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
92       <application path="/test" docBase="E:\work_test\test2" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext">
93     </host>
94     <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
95       channelSendOptions="6">
96
97     <Manager className="com.apusic.ams.ha.session.DeltaManager"
98       expireSessionsOnShutdown="false"
99       notifyListenersOnReplication="true"/>
100
101     <Channel className="com.apusic.ams.tribes.group.GroupChannel">
102       <Membership className="com.apusic.ams.tribes.membership.McastService"
103         address="228.0.0.4"
104         port="45564"
105         frequency="500"
106         dropTime="3000"/>
107       <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
108         address="auto"
109         port="5000"
110         selectorTimeout="100"
111         maxThreads="6"/>
112
113       <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
114         <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
115       </Sender>
116       <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
117       <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
118       <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
119     </Channel>
120
121     <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
122     <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
123   </Cluster>
124 </service>
125 </services>

```

```

84 </endpoints>
85
86 <services>
87   <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
88     <realm className="com.apusic.ams.realm.LockOutRealm">
89       <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
90     </realm>
91     <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
92       <application path="/test" docBase="E:\work_test\test2" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext">
93     </host>
94     <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
95       channelSendOptions="6">
96     <Manager className="com.apusic.ams.ha.session.DeltaManager"
97       expireSessionsOnShutdown="false"
98       notifyListenersOnReplication="true"/>
99     <Channel className="com.apusic.ams.tribes.group.GroupChannel">
100       <Membership className="com.apusic.ams.tribes.membership.McastService"
101         address="228.0.0.4"
102         port="45564"
103         frequency="500"
104         dropTime="3000"/>
105       <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
106         address="auto"
107         port="5000"
108         selectorTimeout="100"
109         maxThreads="6"/>
110       <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
111         <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
112       </Sender>
113       <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
114       <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
115       <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
116     </Channel>
117     <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
118     <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
119     <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
120   </Cluster>
121 </service>
122 </services>
123

```

9.2.1.3 配置 Apache

安裝好 Apache 後，向配置文件 httpd.conf 末尾新增如下內容：

```

Listen 172.x.x.1:8080
ProxyPass / balancer://testcluster/ stickysession=JSESSIONID
ProxyPassReverse / balancer://testcluster/
<Proxy balancer://testcluster>

```

```
BalancerMember http://172.x.x.1:6888/ route=node01
BalancerMember http://172.x.x.1:8888/ route=node02
</Proxy>
```

注意：上面的示例IP端口172.x.x.1:8080 是本地的 Apache 監聽的 IP 地址，需要修改為自己的地址。

172.x.x.1:6888、172.x.x.1:8888 分別是兩個節點的IP，route=xxx 分別填寫上面 service 中的 jvmRoute 的值。

stickySession=JSESSIONID: 會話粘滯，測試時，可先不配置為會話粘滯。

9.2.1.4 驗證叢集

啟動節點 node01:

```
C:\Windows\System32\cmd.exe - apusic.bat run
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
Using APUSIC_BASE: "G:\project\saas\saas\output\build\"
Using APUSIC_HOME: "G:\project\saas\saas\output\build\"
Using APUSIC_TMPDIR: "G:\project\saas\saas\output\build\temp\"
Using JRE_HOME: "F:\tools\jdk1.8_0_201\"
Using CLASSPATH: "G:\project\saas\saas\output\build\bin\bootstrap.jar;G:\project\saas\saas\output\build\bin\saas-juli.jar"
16-Apr-2020 14:52:20.204 [main] com.apusic.ams.startup.VersionLoggerListener.log Server, 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:52:20.209 [main] com.apusic.ams.startup.VersionLoggerListener.log Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:52:20.211 [main] com.apusic.ams.startup.VersionLoggerListener.log 服务器构建号: 10.1.0.0
16-Apr-2020 14:52:20.212 [main] com.apusic.ams.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:52:20.214 [main] com.apusic.ams.startup.VersionLoggerListener.log OS, 版本: 6.1
16-Apr-2020 14:52:20.215 [main] com.apusic.ams.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:52:20.217 [main] com.apusic.ams.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8_0_201\jre
16-Apr-2020 14:52:20.219 [main] com.apusic.ams.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:52:20.221 [main] com.apusic.ams.startup.VersionLoggerListener.log JVM, 供应商: Oracle Corporation
16-Apr-2020 14:52:20.222 [main] com.apusic.ams.startup.VersionLoggerListener.log CATALINA_BASE: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.226 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.229 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=G:\project\saas\saas\output\build\conf\logging.p
16-Apr-2020 14:52:20.232 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=com.apusic.juli.ClassLoaderLogManager
16-Apr-2020 14:52:20.234 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:52:20.236 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:52:20.238 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.ams.webresources
16-Apr-2020 14:52:20.240 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:52:20.242 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.245 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.247 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=G:\project\saas\saas\output\build\temp
16-Apr-2020 14:52:20.249 [main] com.apusic.ams.core.AppLifecycleListener.lifecycleEvent The APR based Apusic AAS Native library which allows optimal performance in production environme
Windows System32\Ntfs;C:\Windows\System32\Ntfs;F:\tools\apache-naven-3.6.0\bin;F:\软件安装\UltraEdit;F:\tools\git\cmd;F:\tools\apache-ant-1.10.5\bin;F:\tools\gradle-5.4.1
16-Apr-2020 14:52:20.494 [main] com.apusic.connector.AbstractProtocol.init 初始化[协议处理器] ("http-nio-6888")
16-Apr-2020 14:52:20.570 [main] com.apusic.ams.startup.Apusic.load 服务器在[651]毫秒内初始化
16-Apr-2020 14:52:20.593 [main] com.apusic.ams.core.StandardService.startInternal Starting service [ams-server]
16-Apr-2020 14:52:20.597 [main] com.apusic.ams.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apusic AAS/10.1.0]
16-Apr-2020 14:52:20.607 [main] com.apusic.ams.http.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:52:20.626 [main] com.apusic.ams.triples.transport.ReceiverBase.bind 服务器直接接收器绑定到: [/172.20.50.43:4000]
16-Apr-2020 14:52:20.636 [main] com.apusic.ams.triples.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:52:20.640 [main] com.apusic.ams.triples.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[4]
16-Apr-2020 14:52:21.643 [main] com.apusic.ams.triples.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[4]
16-Apr-2020 14:52:21.657 [main] com.apusic.ams.triples.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[8]
16-Apr-2020 14:52:22.668 [main] com.apusic.ams.triples.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[8]
16-Apr-2020 14:52:23.491 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用调试日志记录，以获取已扫描但未在其中找到TLD的完整J
16-Apr-2020 14:52:23.505 [main] com.apusic.ams.session.DeltaManager.startInternal 将管理器[localhost#test]注册到名为[ams-service]的集群元素[Engine]
16-Apr-2020 14:52:23.508 [main] com.apusic.ams.session.DeltaManager.startInternal Start clustering manager at [localhost#test]
16-Apr-2020 14:52:23.510 [main] com.apusic.ams.session.DeltaManager.getAllClusterSessions Manager [localhost#test]: skipping state transfer. No members active in cluster group.
16-Apr-2020 14:52:23.533 [main] com.apusic.ams.triples.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context name:/test]
16-Apr-2020 14:52:23.536 [main] com.apusic.ams.triples.tipis.AbstractReplicatedMap.init AbstractReplicatedMap[test] initialization was completed in [3] ms.
16-Apr-2020 14:52:23.540 [main] com.apusic.ams.session.JvmRouteBinderValve.startInternal JvmRouteBinderValve 启动
16-Apr-2020 14:52:23.548 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄("http-nio-6888")
16-Apr-2020 14:52:23.554 [main] com.apusic.ams.startup.Apusic.start Server startup in [2,982] milliseconds
```

啟動節點 node02:

55

```
管理員: C:\Windows\System32\cmd.exe - apusic.bat run
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>apusic.bat run
Using APUSIC_BASE: "G:\project\saas\saas\output\build"
Using APUSIC_HOME: "G:\project\saas\saas\output\build"
Using APUSIC_TMPDIR: "G:\project\saas\saas\output\build\temp"
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "G:\project\saas\saas\output\build\bin\bootstrap.jar;G:\project\saas\saas\output\build\bin\saas-juli.jar"
16-Apr-2020 14:52:20.204 [main] com.apusic.ans.startup.VersionLoggerListener.log Server 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:52:20.209 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器构建: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:52:20.211 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本号: 10.1.0.0
16-Apr-2020 14:52:20.212 [main] com.apusic.ans.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:52:20.214 [main] com.apusic.ans.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:52:20.215 [main] com.apusic.ans.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:52:20.217 [main] com.apusic.ans.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:52:20.219 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:52:20.221 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 供应商: Oracle Corporation
16-Apr-2020 14:52:20.222 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_BASE: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.226 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_HOME: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.229 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=G:\project\saas\saas\output\build\conf\logging.p
16-Apr-2020 14:52:20.232 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=com.apusic.juli.ClassLoaderLogManager
16-Apr-2020 14:52:20.234 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:52:20.236 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:52:20.238 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.ans.webresources
16-Apr-2020 14:52:20.240 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:52:20.242 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.245 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.247 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=G:\project\saas\saas\output\build\temp
16-Apr-2020 14:52:20.249 [main] com.apusic.ans.core.ApplifecycleListener.lifecycleEvent The APR based Apusic AAS Native library which allows optimal performance in production environme
16-Apr-2020 14:52:20.250 [main] com.apusic.connector.AbstractProtocol.init 初始化协议处理器 ("http-nio-6888")
16-Apr-2020 14:52:20.250 [main] com.apusic.ans.core.StandardService.startInternal 服务器在6888端口初始化
16-Apr-2020 14:52:20.592 [main] com.apusic.ans.core.StandardEngine.startInternal Starting service [ans-server]
16-Apr-2020 14:52:20.607 [main] com.apusic.ans.ha.tcp.SimpleTcpCluster.startInternal 正在启动 Servlet 引擎, [Apusic AAS/10.1.0]
16-Apr-2020 14:52:20.626 [main] com.apusic.ans.ha.tcp.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:52:20.636 [main] com.apusic.ans.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4000]
16-Apr-2020 14:52:20.640 [main] com.apusic.ans.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:52:20.643 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership. start level:[4]
16-Apr-2020 14:52:21.643 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping. membership established. start level:[4]
16-Apr-2020 14:52:21.657 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership. start level:[8]
16-Apr-2020 14:52:22.657 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping. membership established. start level:[8]
16-Apr-2020 14:52:22.668 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping. membership established. start level:[8]
16-Apr-2020 14:52:23.491 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。 为此记录器启用调试日志记录, 以获取已扫描但未在其中找到TLD的完整J
16-Apr-2020 14:52:23.505 [main] com.apusic.ans.ha.session.DeltaManager.startInternal 将管理器 [localhost#test]注册到名为 [ans-service]的集群元素 [Engine]
16-Apr-2020 14:52:23.508 [main] com.apusic.ans.ha.session.DeltaManager.startInternal Starting clustering manager at [localhost#test]
16-Apr-2020 14:52:23.510 [main] com.apusic.ans.ha.session.DeltaManager.getAllClusterSessions Manager [localhost#test]: skipping state transfer. No members active in cluster group.
16-Apr-2020 14:52:23.533 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context name:[test]
16-Apr-2020 14:52:23.536 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init AbstractReplicatedMap [test] initialization was completed in [3] ms.
16-Apr-2020 14:52:23.540 [main] com.apusic.ans.ha.session.JvmRouteBinderValve.startInternal JvmRouteBinderValve 启动
16-Apr-2020 14:52:23.548 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄 ["http-nio-6888"]
16-Apr-2020 14:52:23.554 [main] com.apusic.ans.startup.Apusic.start Server startup in [2,982] milliseconds
16-Apr-2020 14:52:57.706 [Tribe-Task-Receiver[ans-service-Channel]-1] com.apusic.ans.tribes.io.BufferPool.getBufferPool Created a buffer pool with max size:[104857600] bytes of type:
16-Apr-2020 14:52:58.506 [Membership-MemberAdded.1] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberAdded Replication member added:[com.apusic.ans.tribes.membership.MemberImpl(tcp://172.
16-Apr-2020 14:54:21.286 [Tribe-Task-Receiver[ans-service-Channel]-61] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员[com.apusic.ans.tribes.membership.MemberImpl(tcp://172.
16-Apr-2020 14:54:21.287 [Tribe-Task-Receiver[ans-service-Channel]-61] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared map 条目的重定位在 [0] ms内完成。
16-Apr-2020 14:54:21.309 [Membership-MemberDisappeared.1] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberDisappeared 收到成员消失:[com.apusic.ans.tribes.membership.MemberImpl(tcp://172.
16-Apr-2020 14:54:25.579 [Membership-MemberAdded.1] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberAdded Replication member added:[com.apusic.ans.tribes.membership.MemberImpl(tcp://172.
```

訪問 <http://172.x.x.1:8080/test/session.jsp> 多重新整理幾次:

```
管理員: C:\Windows\System32\cmd.exe - apusic.bat run
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "G:\project\saas\saas\output\build\bin\bootstrapper.jar;G:\project\saas\saas\output\build\bin\saas-juli.jar"
16-Apr-2020 14:52:20.204 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:52:20.209 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:52:20.211 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本: 10.1.0.0
16-Apr-2020 14:52:20.212 [main] com.apusic.ans.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:52:20.214 [main] com.apusic.ans.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:52:20.215 [main] com.apusic.ans.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:52:20.217 [main] com.apusic.ans.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:52:20.219 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:52:20.221 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 供应商: Oracle Corporation
16-Apr-2020 14:52:20.222 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_BASE: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.226 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_HOME: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.232 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=G:\project\saas\saas\output\build\conf\logging.p
16-Apr-2020 14:52:20.234 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=com.apusic.juli.ClassLoaderLogManager
16-Apr-2020 14:52:20.236 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:52:20.238 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:52:20.240 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.ans.webresources
16-Apr-2020 14:52:20.242 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.245 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.247 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=G:\project\saas\saas\output\build\temp
16-Apr-2020 14:52:20.249 [main] com.apusic.ans.core.AprLifecycleListener.lifecycleEvent The APR based Apusic AAS Native library which allows optimal performance in production environme
Windows\System32\WindowsPowerShell\1.0\cmd;F:\tools\apache-maven-3.6.0\bin;F:\软件安装\UltraEdit;F:\tools\Git\cmd;F:\tools\apache-ant-1.10.5\bin;F:\tools\gradle-5.4.1
16-Apr-2020 14:52:20.494 [main] com.apusic.connector.AbstractProtocol.init 初始化协议处理器 ["http-nio-6888"]
16-Apr-2020 14:52:20.570 [main] com.apusic.ans.startup.Apusic.load 服务器在6511毫秒内初始化
16-Apr-2020 14:52:20.593 [main] com.apusic.ans.core.StandardService.startInternal Starting service [ans-server]
16-Apr-2020 14:52:20.597 [main] com.apusic.ans.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apusic AAS/10.1.0]
16-Apr-2020 14:52:20.607 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:52:20.626 [main] com.apusic.ans.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4000]
16-Apr-2020 14:52:20.636 [main] com.apusic.ans.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:52:20.640 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[4]
16-Apr-2020 14:52:21.643 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[4]
16-Apr-2020 14:52:21.657 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[8]
16-Apr-2020 14:52:22.668 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[8]
16-Apr-2020 14:52:23.491 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用调试日志记录,以获取已扫描但未在其中找到TLD的完整J
16-Apr-2020 14:52:23.505 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal 将管理器 [localhost#test]注册到名为 [ans-service]的集群元素 [Engine]
16-Apr-2020 14:52:23.510 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Starting clustering manager at [localhost#test]
16-Apr-2020 14:52:23.518 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Skipping state transfer. No members active in cluster group.
16-Apr-2020 14:52:23.533 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context name:[/test]
16-Apr-2020 14:52:23.536 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init AbstractReplicatedMap[/test] initialization was completed in [3] ms.
16-Apr-2020 14:52:23.540 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal JvmRouteBinderValve 启动
16-Apr-2020 14:52:23.548 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄 ["http-nio-6888"]
16-Apr-2020 14:52:23.554 [main] com.apusic.ans.startup.Apusic.start Server startup in [2,982] milliseconds
16-Apr-2020 14:52:57.706 [Tribe-Task-Receiver[ans-service-Channel]-1] com.apusic.ans.tribes.io.BufferPool.getBufferPool Created a buffer pool with max size:[104857600] bytes of type:
16-Apr-2020 14:52:58.506 [Membership-MemberAdded.1] com.apusic.ans.tcp.SimpleTcpCluster.memberAdded Replication member added: [com.apusic.ans.tribes.membership.MemberImpl[http://172.
16-Apr-2020 14:54:21.286 [Tribe-Task-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员 [com.apusic.ans.tribes.membership.MemberImpl[http://172.
16-Apr-2020 14:54:21.287 [Tribe-Task-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员 [com.apusic.ans.tribes.membership.MemberImpl[http://172.
16-Apr-2020 14:54:21.309 [Membership-MemberDisappeared.1] com.apusic.ans.tcp.SimpleTcpCluster.memberDisappeared 收到成员消失: [com.apusic.ans.tribes.membership.MemberImpl[http://172.
16-Apr-2020 14:54:25.529 [Membership-MemberAdded.1] com.apusic.ans.tcp.SimpleTcpCluster.memberAdded Replication member added: [com.apusic.ans.tribes.membership.MemberImpl[http://172.
16-Apr-2020 14:55:56.558 [http-exec-11] org.apache.jsp.session_jsp._jspService session put: 2020-04-16 14:55:56
16-Apr-2020 14:56:00.105 [http-exec-21] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.082 [http-exec-41] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.429 [http-exec-51] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.766 [http-exec-61] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.122 [http-exec-71] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.460 [http-exec-81] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
```

支援使用 Redis 進行 Session 集中式儲存。

9.2.2.1.1 系統構建

```
common.loader="${apusic.base}/lib","${apusic.base}/lib/*.jar","${apusic.home}/lib","${apusic.home}/lib/*.jar"
```

```
<Valve className="com.apusic.ams.session.redissessions.RedisSessionHandlerValve" />
<Manager className="com.apusic.ams.session.redissessions.RedisSessionManager" host="localhost"
```

```
port="6379" database="0" />
```

如果 Redis 使用了主從、哨兵、叢集模式，`${APUSIC_HOME}/conf/context.xml` 中的 `Manager` 配置對應修改。如 Redis 使用了哨兵模式，`Manager` 配置如下，IP端口為示例，需要根據實際修改：

```
<Manager className="com.apusic.ams.session.redissessions.RedisSessionManager" password="[YOUR_PASSWORD]"
sentinelMaster="mymaster"
    sentinels="172.x.x.1:26379,172.x.x.2:26379" database="0" />
```

表37：金蝶Apusic應用伺服器V10標準版session集中式配置Manager標籤說明

引數	說明
host	Redis 伺服器地址
port	端口
sentinelMaster	Redis server 名字
sentinels	哨兵, 用逗號分割開的“host:port”
password	Redis 密碼
blockWhenExhausted	連線耗盡時是否阻塞, false 報異常,true 阻塞直到超時, 默認 true
connectionPoolMaxIdle	資源池允許最大空間的連線數, 默認 8
connectionPoolMaxTotal	資源池中最大連線數, 默認 8
connectionPoolMinIdle	資源池確保最少空間的連線數, 默認 0
database	DB 索引
jmxEnabled	是否啟用 pool 的 jmx 管理功能, 默認 true
lifo	是否啟用後進先出, 默認 true
maxWaitMillis	獲取連線時的最大等待毫秒數 (如果設置為阻塞時 BlockWhenExhausted), 如果超時就拋異常, 小於零 : 阻塞不確定的時間, 默認 -1
minEvictableIdleTimeMillis	逐出連線的最小空間時間 默認 1800000 毫秒 (30 分鐘)
numTestsPerEvictionRun	每次逐出檢查時 逐出的最大數目 如果為負數就是 : 1/abs(n), 默認 3
sessionPersistPolicies	“SAVE_ON_CHANGE” : 每次調用 session.setAttribute() 或者 session.removeAttribute() 時, Session 都會儲存到 Redis。 “ALWAYS_SAVE_AFTER_REQUEST”: 請求結束後, 儲存 Session, 不管是否檢測到 Session 屬性變化。 “DEFAULT”: 默認。當檢測到 Session 屬性變化時, 才儲存
testOnBorrow	向資源池借用連線時是否做連線有效性檢測 (ping), 無效連線會被移除, 默認 false
testOnCreate	向資源池建立連線時是否做連線有效性檢測 (ping)
testOnReturn	向資源池歸還連線時是否做連線有效性檢測 (ping), 無效連線會被移除, 默認 false
testWhileIdle	在空間時檢查有效性, 默認 false
timeBetweenEvictionRunsMillis	逐出掃描的時間間隔 (毫秒) 如果為負數, 則不執行逐出線程, 默認 -1
timeout	讀取超時

10 嵌入式部署（與 Spring Boot 整合使用）

10.1 前言

本部分為金蝶 Apusic 應用伺服器V10嵌入式部署使用說明，詳細介紹金蝶 Apusic 應用伺服器V10嵌入式部署安裝和相關配置方法。

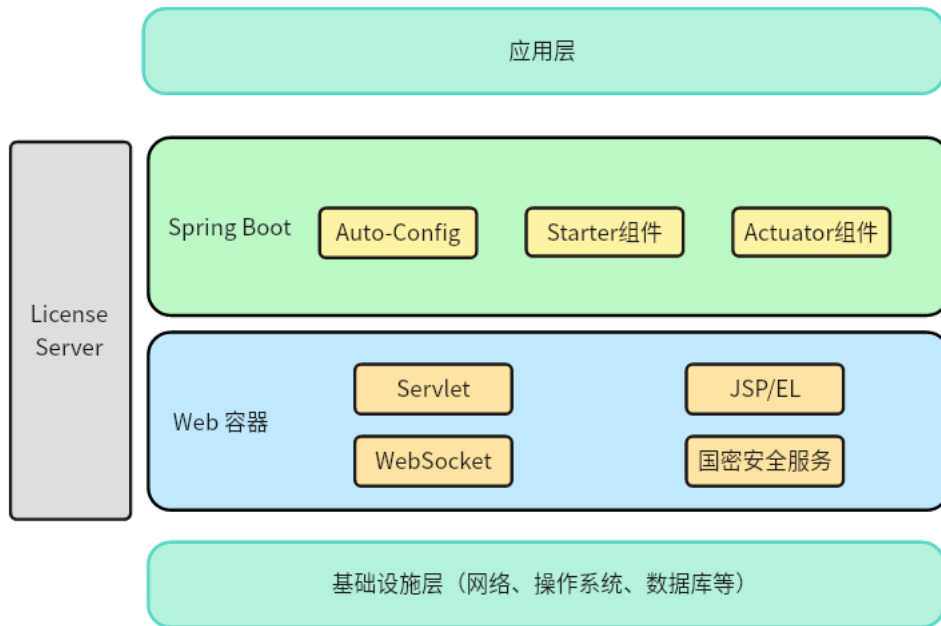
10.1.1 面向對象

本部分主要面向對象為使用金蝶 Apusic 應用伺服器V10嵌入式部署進行應用開發的開發人員，以及相關管理人員和運維人員。

10.1.2 概述

金蝶 Apusic 應用伺服器V10嵌入式部署是一款同時支援 JavaEE/JakartaEE 企業級標準與 React 響應程式設計標準的嵌入式應用伺服器，整合Spring Boot框架使用。

圖2：金蝶Apusic應用伺服器V10嵌入式部署架構圖



10.1.3 產品規範與環境要求

嵌入式部署的產品規範（JakartaEE 規範）、平台環境與系統環境要求與獨立部署一致，參見「產品概述-產品簡介」。嵌入式部署使用 `AAMS-[version]-embed` 產品包（金蝶 Apusic應用伺服器嵌入式部署產品標識）。

10.2 產品清單

金蝶Apusic應用伺服器V10嵌入式部署產品包說明，主要包括starter、websocket、國密等資源包。詳看產品清單說明。

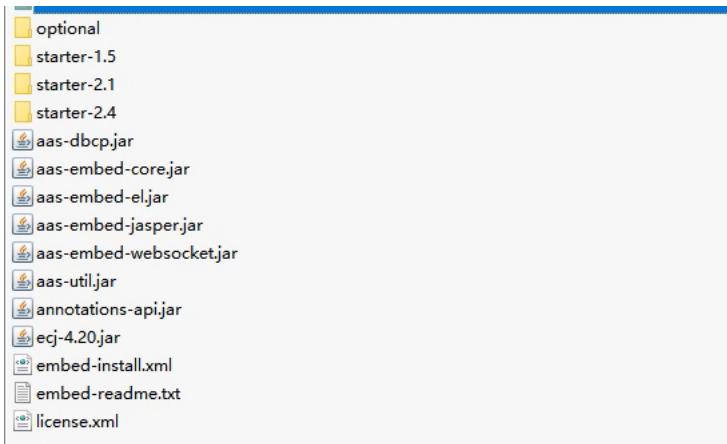
表38：金蝶Apusic應用伺服器V10嵌入式部署產品包

產品包	說明
aams-spring-boot-starter-all-[version].jar	AAMS starter
aams-spring-boot-websocket-starter-[version].jar	WebSocket資源包
aas-sm-[version].jar	國密資源包

10.2.1 產品清單列表

金蝶Apusic應用伺服器V10嵌入式部署產品包包括starter、相關資源模組、可選資源模組等內容。

圖3：金蝶Apusic應用伺服器V10嵌入式部署產品清單列表



10.2.2 產品清單說明

金蝶Apusic應用伺服器V10嵌入式部署產品清單說明。

表39：金蝶Apusic應用伺服器V10嵌入式部署產品清單說明

產品包	說明
optional	可選元件，包括aas-sm-3.80.1.jar（國密模組）和acls-client-1.0.9.jar（集中授權模組）。
starter-1.5	Spring boot 1.x版本的starter，包括以下文件： aams-spring-boot-starter-all-1.5.20.RELEASE.jar：AAS的starter-all模組，其中包含了starter和websocket-starter模組 aams-spring-boot-websocket-starter-1.5.20.RELEASE.jar：AAS的websocket-starter模組 aams-spring-boot-starter-1.5.20.RELEASE.jar：AAS的starter模組 aams-spring-boot-websocket-starter.pom.xml：AAS的websocket-starter模組的maven配置文件 aams-spring-boot-starter.pom.xml：AAS的starter模組的maven配置文件
starter-2.1	Spring boot 2.0.0~2.3.x版本的starter，包括以下文件： aams-spring-boot-starter-all-2.1.7.RELEASE.jar：AAS的starter-all模組，其中包含了starter、web-starter和websocket-starter模組 aams-spring-boot-websocket-starter-2.1.7.RELEASE.jar：AAS的websocket-starter模組 aams-spring-boot-starter-2.1.7.RELEASE.jar：AAS的starter模組 aams-spring-boot-web-starter-2.1.7.RELEASE.jar：AAS的web-starter模組 aams-spring-boot-websocket-starter.pom.xml：AAS的websocket-starter模組的maven配置文件 aams-spring-boot-starter.pom.xml：AAS的starter模組的maven配置文件 aams-spring-boot-web-starter.pom.xml：AAS的web-starter模組的maven配置文件
starter-2.4	Spring boot 2.4.0及以上版本的starter，包括以下文件： aams-spring-boot-starter-all-2.4.0.jar：AAS的starter-all模組，其中包含了starter、web-starter和websocket-starter模組 aams-spring-boot-websocket-starter-2.4.0.jar：AAS的websocket-starter模組 aams-spring-boot-starter-2.4.0.jar：AAS的starter模組 aams-spring-boot-web-starter-2.4.0.jar：AAS的web-starter模組 aams-spring-boot-websocket-starter.pom.xml：AAS的websocket-starter模組的maven配置文件 aams-spring-boot-starter.pom.xml：AAS的starter模組的maven配置文件 aams-spring-boot-web-starter.pom.xml：AAS的web-starter模組的maven配置文件
aas-dbc.jar aas-embed-core.jar aas-embed-el.jar aas-embed-jasper.jar aas-embed-websocket.jar aas-util.jar annotations-api.jar ecj-4.20.jar	AAS相關模組
embed-install.xml	AAS相關模組的快捷式引入文件，命令：mvn initialize -f embed-install.xml
embed-readme.txt	簡要安裝說明
license.xml	授權文件

10.3 安裝

安裝方式有兩種：一種是基於應用系統原始碼重新構建可執行jar包，另一種是基於可執行jar包進行替換。

需注意，匯入jar包時需根據每個jar包的實際情況修改具體的jar名。

10.3.1 基於原始碼方式構建

為了方便與應用快速整合，金蝶Apusic應用伺服器V10嵌入式部署的使用方式共有兩種，第一種是隻需匯入單個jar，該jar是包含了金蝶Apusic應用伺服器V10嵌入式部署和starter相關的jar，另一種是按模組匯入jar的方式，可將金蝶Apusic應用伺服器V10嵌入式部署、starter分模組匯入，產品包提供了各版本的starter以及可選的模組，下面分別介紹這兩種使用方式。

10.3.1.1 匯入單個jar

為了方便與應用快速整合，應用伺服器支援匯入單個jar，該jar包含了aas和starter相關的jar。通常命名為 `aams-spring-boot-starter-all-[version].jar`。

10.3.1.1.1 前置條件

安裝maven環境；

安裝java環境；

需要下載產品包 `AAMS-[version]-embed` 到本地。

10.3.1.1.2 安裝資源依賴

`aams-spring-boot-starter-all`

解壓 `AAMS-[version]-embed` 通過終端進入 `AAMS-[version]-embed` 目錄。

springboot1.x版本:

通過終端進入 `AAMS-[version]-embed/starter-1.5` 目錄，輸入以下命令匯入jar包。

```
mvn install:install-file -Dfile=aams-spring-boot-starter-all-1.5.20.RELEASE.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-starter-all -Dversion=1.5.20.RELEASE -Dpackaging=jar
```

```

\starter-1.5>mvn install:install-file -Dfile=aams-spring-boot-starter-all-1.5.20.RELEASE.jar -
DgroupId=com.apusic -DartifactId=aams-spring-boot-starter-all -Dversion=1.5.20.RELEASE -Dpackaging=jar
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO]
[INFO] --- install:3.1.0:install-file (default-cli) @ standalone-pom ---
[INFO] pom.xml not found in aams-spring-boot-starter-all-1.5.20.RELEASE.jar
[INFO] Installing \starter-1.5\
aams-spring-boot-starter-all-1.5.20.RELEASE.jar to D:\software\
mvn\repository\com\apusic\
aams-spring-boot-starter-all\1.5.20.RELEASE\
aams-spring-boot-starter-all-1.5.20.RELEASE.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.348 s
[INFO] Finished at: 2024-05-28T09:51:18+08:00
[INFO]

```

springboot2.0.0~springboot2.3.x版本:

通過終端進入 `AAMS-[version]-embed/starter-2.1` 目錄，輸入以下命令匯入jar包。

```
mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.1.7.RELEASE.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-starter-all -Dversion=2.1.7.RELEASE -Dpackaging=jar
```

```

... \starter-2.1>mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.1.7.RELEASE.jar -D
groupId=com.apusic -DartifactId=aams-spring-boot-starter-all -Dversion=2.1.7.RELEASE -Dpackaging=jar
[INFO] Scanning for projects...
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO] --- install:3.1.0:install-file (default-cli) @ standalone-pom ---
[INFO] pom.xml not found in aams-spring-boot-starter-all-2.1.7.RELEASE.jar
[INFO] Installing I... \starter-2.1\ aams-spring-boot-starter-all-2.1.7.RELEASE.jar to D:\software\m
vn\repository\com\apusic\ aams-spring-boot-starter-all\2.1.7.RELEASE\ aams-spring-boot-starter-all-2.1.7.RELEASE.jar
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.365 s
[INFO] Finished at: 2024-05-28T09:52:58+08:00
[INFO] -----

```

springboot2.4.0及以上版本:

通過終端進入 AAMS-[version]-embed/starter-2.4 目錄，輸入以下命令匯入jar包。

```

mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.4.0.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-starter-all -Dversion=2.4.0 -Dpackaging=jar

```

```

... \starter-2.4>mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.4.0.jar -DgroupId=
com.apusic -DartifactId=aams-spring-boot-starter-all -Dversion=2.4.0 -Dpackaging=jar
[INFO] Scanning for projects...
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO] --- install:3.1.0:install-file (default-cli) @ standalone-pom ---
[INFO] pom.xml not found in aams-spring-boot-starter-all-2.4.0.jar
[INFO] Installing ... \starter-2.4\ aams-spring-boot-starter-all-2.4.0.jar to D:\software\mvn\respo
sitory\com\apusic\ aams-spring-boot-starter-all\2.4.0\ aams-spring-boot-starter-all-2.4.0.jar
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.368 s
[INFO] Finished at: 2024-05-28T10:18:52+08:00
[INFO] -----

```

10.3.1.1.3 引入資源依賴

aams-spring-boot-starter

使用金蝶Apusic應用伺服器V10嵌入式部署作為執行容器，可以在 pom.xml 中新增 aams-spring-boot-starter 和金蝶Apusic應用伺服器V10嵌入式部署的依賴，同時排除 tomcat 的依賴。

由於 spring-boot-starter-web 外掛默認帶了 tomcat 的 starter，所以在 pom.xml 中需要排除 tomcat 的 starter，如下面的配置：

```

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>

```

aams-spring-boot-starter-all

增加金蝶Apusic應用伺服器V10嵌入式部署的 starter 依賴，如 pom.xml 中的如下描述：

如果springboot版本是 1.x 版本，則 pom 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter-all</artifactId>
  <version>1.5.20.RELEASE</version>
</dependency>
```

如果springboot版本是 springboot2.0.0-springboot2.3.x 版本，則 pom 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter-all</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
```

如果springboot版本是 springboot2.4.0 以上版本，則 pom 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter-all</artifactId>
  <version>2.4.0</version>
</dependency>
```

10.3.1.1.4 驗證安裝

訪問 maven 的本地倉庫地址" `.../repository/com/apusic/` "，可以在對應倉庫地址 中找到上述的 jar 包，此時即表示 jar 包部署到本地倉庫成功。

加卷: repository > com > apusic >

名称	修改日期	类型
aams-spring-boot-starter	2024/5/22 17:57	文件夹
aams-spring-boot-starter-all	2024/5/23 9:36	文件夹
aams-spring-boot-web-starter	2023/6/5 11:05	文件夹
aas	2023/12/1 12:06	文件夹
aas-annotations-api	2024/5/22 17:58	文件夹
aas-dbcsp	2024/5/22 15:29	文件夹
aas-embed-core	2024/5/22 17:57	文件夹
aas-embed-el	2024/5/22 17:58	文件夹
aas-embed-jasper	2024/5/22 15:29	文件夹
aas-embed-websocket	2024/5/22 17:58	文件夹
aas-juli	2024/5/22 15:29	文件夹
aas-sm	2023/12/1 12:06	文件夹
aas-util	2024/5/22 15:29	文件夹
acls-client	2024/5/22 15:29	文件夹
ams	2023/6/5 11:03	文件夹

10.3.1.2 按模組匯入方式

應用伺服器同時支援按模組匯入jar的方式，可將aas、starter分模組匯入，產品包提供了各版本的starter以及可選的模組。

10.3.1.2.1 前置條件

安裝maven環境；

安裝java環境；

需要下載產品包 AAMS-[version]-embed 到本地；

AAMS-[version]-embed 目錄中 embed-install.xml 存在。

10.3.1.2.2 安裝資源依賴

aas嵌入式版模組

通過終端進入 `embed-install.xml` 同級目錄執行：`mvn initialize -f embed-install.xml` 或以下命令：

```
mvn install:install-file -Dfile=aas-embed-core.jar -DgroupId=com.apusic -DartifactId=aas-embed-core -
Dversion=10.0.0 -Dpackaging=jar
mvn install:install-file -Dfile=aas-embed-el.jar -DgroupId=com.apusic -DartifactId=aas-embed-el -
Dversion=10.0.0 -Dpackaging=jar
mvn install:install-file -Dfile=aas-embed-websocket.jar -DgroupId=com.apusic -DartifactId=aas-embed-
websocket -Dversion=10.0.0 -Dpackaging=jar
mvn install:install-file -Dfile=aas-embed-jasper.jar -DgroupId=com.apusic -DartifactId=aas-embed-jasper -
Dversion=10.0.0 -Dpackaging=jar
mvn install:install-file -Dfile=aas-dbcj.jar -DgroupId=com.apusic -DartifactId=aas-dbcj -Dversion=10.0.0 -
Dpackaging=jar
mvn install:install-file -Dfile=aas-util.jar -DgroupId=com.apusic -DartifactId=aas-util -Dversion=10.0.0 -
Dpackaging=jar
mvn install:install-file -Dfile=ecj-4.20.jar -DgroupId=com.apusic -DartifactId=ecj -Dversion=4.20 -
Dpackaging=jar
mvn install:install-file -Dfile=optional/aas-sm-3.80.1.jar -DgroupId=com.apusic -DartifactId=aas-sm -
Dversion=3.80.1 -Dpackaging=jar
mvn install:install-file -Dfile=optional/acls-client-1.0.9.jar -DgroupId=com.apusic -DartifactId=acls-
client -Dversion=1.0.9 -Dpackaging=jar
```

其中 `optional/aas-sm-3.80.1.jar` 是支援國密必選模組，`optional/acls-client-1.0.9.jar` 是支援集中授權必選模組。

starter模組

接下來匯入對應版本的starter。

springboot1.x版本:

通過終端進入 `AAMS-[version]-embed/starter-1.5` 目錄，輸入以下命令匯入jar包。

```
mvn install:install-file -Dfile=aams-spring-boot-starter-1.5.20.RELEASE.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-starter -Dversion=1.5.20.RELEASE -Dpackaging=jar -DpomFile=aams-spring-boot-
starter.pom.xml
mvn install:install-file -Dfile=aams-spring-boot-websocket-starter-1.5.20.RELEASE.jar -DgroupId=com.apusic
-DartifactId=aams-spring-boot-websocket-starter -Dversion=1.5.20.RELEASE -Dpackaging=jar -DpomFile=aams-
spring-boot-websocket-starter.pom.xml
```

springboot2.0.0~springboot2.3.x版本:

通過終端進入 `AAMS-[version]-embed/starter-2.1` 目錄，輸入以下命令匯入jar包。

```
mvn install:install-file -Dfile=aams-spring-boot-starter-2.1.7.RELEASE.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-starter -Dversion=2.1.7.RELEASE -Dpackaging=jar -DpomFile=aams-spring-boot-
starter.pom.xml
mvn install:install-file -Dfile=aams-spring-boot-websocket-starter-2.1.7.RELEASE.jar -DgroupId=com.apusic
-DartifactId=aams-spring-boot-websocket-starter -Dversion=2.1.7.RELEASE -Dpackaging=jar -DpomFile=aams-
spring-boot-websocket-starter.pom.xml
mvn install:install-file -Dfile=aams-spring-boot-web-starter-2.1.7.RELEASE.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-web-starter -Dversion=2.1.7.RELEASE -Dpackaging=jar -DpomFile=aams-spring-
boot-web-starter.pom.xml
```

springboot2.4.0及以上版本:

通過終端進入 `AAMS-[version]-embed/starter-2.4` 目錄，輸入以下命令匯入jar包。

```
mvn install:install-file -Dfile=aams-spring-boot-starter-2.4.0.jar -DgroupId=com.apusic -DartifactId=aams-
spring-boot-starter -Dversion=2.4.0 -Dpackaging=jar -DpomFile=aams-spring-boot-starter.pom.xml
```

```
mvn install:install-file -Dfile=aams-spring-boot-websocket-starter-2.4.0.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-websocket-starter -Dversion=2.4.0 -Dpackaging=jar -DpomFile=aams-spring-boot-
websocket-starter.pom.xml
mvn install:install-file -Dfile=aams-spring-boot-web-starter-2.4.0.jar -DgroupId=com.apusic -
DartifactId=aams-spring-boot-web-starter -Dversion=2.4.0 -Dpackaging=jar -DpomFile=aams-spring-boot-web-
starter.pom.xml
```

10.3.1.2.3 引入資源依賴

在專案的 `pom.xml` 文件中匯入對應的模組。

如果springboot版本是 1.x 版本，則 `pom` 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter</artifactId>
  <version>1.5.20.RELEASE</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-websocket-starter</artifactId>
  <version>1.5.20.RELEASE</version>
</dependency>
```

如果springboot版本是 `springboot2.0.0-springboot2.3.x` 版本，則 `pom` 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-websocket-starter</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-web-starter</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
```

如果springboot版本是 `springboot2.4.0` 以上版本，則 `pom` 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter</artifactId>
  <version>2.4.0</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-websocket-starter</artifactId>
  <version>2.4.0</version>
</dependency>
<dependency>
```

```
<groupId>com.apusic</groupId>
<artifactId>aams-spring-boot-web-starter</artifactId>
<version>2.4.0</version>
</dependency>
```

匯入 starter 之後，應用的 pom.xml 需要排除tomcat的 starter，參考如下：

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

10.3.1.2.4 驗證安裝

訪問 maven 的本地倉庫地址 "`.../repository/com/apusic/`"，可以在對應倉庫地址 中找到上述的 jar 包，此時即表示 jar 包部署到本地倉庫成功。

10.3.2 基於可執行jar替換

若應用已經編譯為可執行資源 jar 包，且採用的是 Spring Boot 內建的 Tomcat 伺服器或其他應用伺服器，需要把 Tomcat 嵌入式或其他應用伺服器嵌入式的 jar 包刪除，然後根據不同的 Spring Boot 版本把 金蝶Apusic應用伺服器V10嵌入式部署所需的 jar 包新增進去。

10.3.2.1 前置條件

應用專案已開發完成，且編譯為可執行資源jar包。

10.3.2.2 刪除棄用依賴

以Tomcat嵌入式應用為例。

使用rar 或 7zip 等壓縮工具開啟已編譯完成的 jar 包。

進入 "`BOOT-INF/lib`" 目錄，刪除如下jar包。

```
tomcat-embed-core-x.x.x.jar
tomcat-embed-el-x.x.x.jar
tomcat-embed-websocket-x.x.x.jar
tomcat-embed-jasper-x.x.x.jar
spring-boot-starter-tomcat-x.x.x.jar
```

10.3.2.3 引入資源依賴

10.3.2.3.1 匯入單個JAR

使用rar 或 7zip 等壓縮工具開啟已編譯完成的 jar 包。

進入 "`BOOT-INF/lib`" 目錄，新增如下jar包。

如果springboot版本是 1.x 版本，則新增 `aams-spring-boot-starter-all-1.5.20.RELEASE.jar` 包。

如果springboot版本是 `springboot2.0.0-springboot2.3.x` 版本，則新增 `aams-spring-boot-starter-all-2.1.7.RELEASE.jar` 包。

如果springboot版本是 `springboot2.4.0` 及以上版本，則新增 `aams-spring-boot-starter-all-2.4.0.jar` 包。

10.3.2.3.2 按模組匯入

使用rar 或 7zip 等壓縮工具開啟已編譯完成的 jar 包。

進入 "`BOOT-INF/lib`" 目錄，新增如下jar包。

如果springboot版本是 1.x 版本，則新增 `aams-spring-boot-starter-1.5.20.RELEASE.jar` 包。

如果springboot版本是 `springboot2.0.0-springboot2.3.x` 版本，則新增 `aams-spring-boot-starter-2.1.7.RELEASE.jar` 和 `aams-spring-boot-web-starter-2.1.7.RELEASE.jar` 包。

如果springboot版本是 `springboot2.4.0` 及以上版本，則新增 `aams-spring-boot-starter-2.4.0.jar` 和 `aams-spring-boot-web-starter-2.4.0.jar` 包。

若原專案中用到了 websocket 規範，在刪除了包 `tomcat-embed-websocket-x.x.x.jar` 之後，還需要放入如下 jar 包：

springboot版本是 `1.x` 版本：`aams-spring-boot-websocket-starter-1.5.20.RELEASE.jar`

springboot版本是 `2.0.0-2.3.x` 版本：`aams-spring-boot-websocket-starter-2.1.7.RELEASE.jar`

springboot版本是 `2.4.0` 及以上版本：`aams-spring-boot-websocket-starter-2.4.0.jar`

10.4 授權認證

金蝶 Apusic 應用伺服器嵌入式部署啟動需要依賴授權文件進行校驗，一般情況下產品自帶有效授權文件，如果授權文件將要過期或者已經過期，可聯絡金蝶天燕對接人員申領授權文件（`license.xml`），與此同時需要將產品特徵碼發給對接人員。授權認證方式有：專案目錄啟動（未打包狀態）、引數設置啟動（未打包狀態）、專案目錄啟動（可執行jar包狀態）、引數設置啟動（可執行jar包狀態），以及集中授權。

10.4.1 獲取特徵碼

在與springboot整合的方式下，沒有獲取特徵碼的 `apusic` 命令，需要依賴用戶的應用進行獲取。

如果已經有臨時的授權文件可以啟動應用，則先把授權文件移除，然後在啟動命令中增加JVM引數 `apusic-ac`，取值為ip地址或者網絡卡名稱，類似如下，eth3為測試機器的網絡卡名稱，需要根據實際的網路資訊修改：

```
java -Dapusic-ac=eth3 -jar boot-quickstart-1.0.jar
```

由於沒有授權文件，應用啟動會停止，在停止前會列印特徵碼資訊，類似如下，Auth Code=特徵碼內容：

```
Auth Code=SZTY263537626
license is invalid.
```

獲取特徵碼後再提供特徵碼申請授權文件。

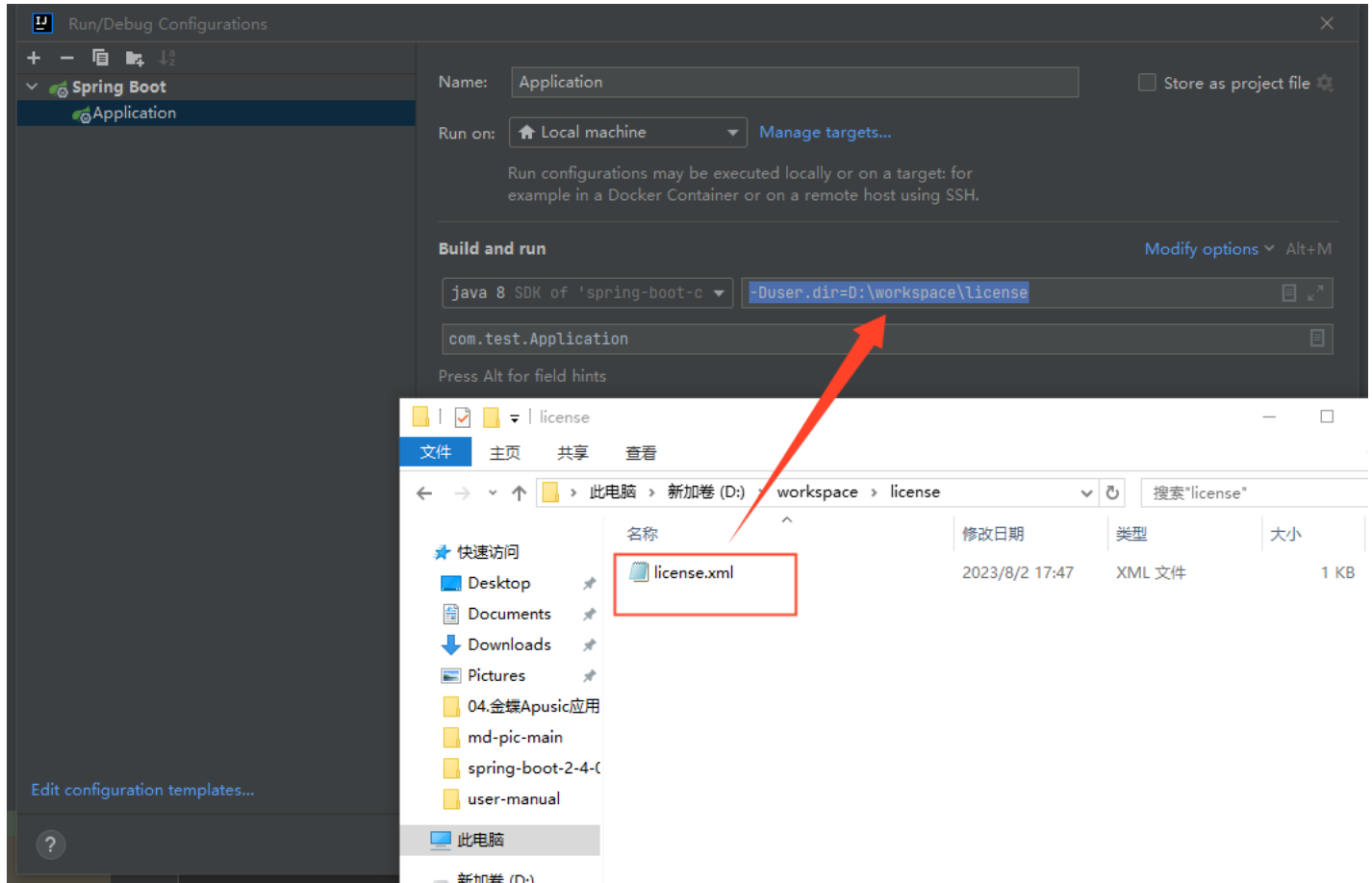
10.4.2 專案目錄啟動（未打包狀態）

將 `license.xml` 文件放在專案啟動文件目錄下，如下圖所示。

名称	修改日期	类型	大小
.idea	2023/8/21 14:54	文件夹	
LOG_FILE_PATH_IS_UNDEFINED	2023/8/21 14:54	文件夹	
src	2023/8/16 10:13	文件夹	
target	2023/8/16 13:35	文件夹	
license.xml	2023/8/2 17:47	XML 文件	1 KB
pom.xml	2023/8/17 16:41	XML 文件	6 KB

10.4.3 引數設置啟動（未打包狀態）

開啟idea，找到Run->Edit Configuration->Spring Boot中的專案啟動類，在右邊的VM options中輸入 `-Duser.dir=xxxxxxx`，`xxxxxxx` 表示 `license.xml` 文件所在的目錄，如下圖所示。



10.4.4 專案目錄啟動（可執行jar包狀態）

將 `license.xml` 文件放在專案啟動文件目錄下，如下圖所示。

/opt/AAS/		
Name	Size (KB)	Last modified
..		
springboot-helloworld-0.0.1-S...	23 536	2024-09-23 14:35
license.xml	1	2024-09-23 14:36

10.4.5 引數設置啟動（可執行jar包狀態）

當專案已經打包為可執行jar包狀態時，可以通過命令對專案進行授權認證。

```
java -Dlicpath=[license.xml文件的絕對路徑] -jar [可執行jar包]
```

10.4.6 集中授權

應用伺服器支援集中授權，通過連線授權中心，由授權中心統一下發管理授權文件，實現集中授權。

需要將集中授權模組匯入到本地庫。

匯入 jar 文件 `acls-client-1.0.9.jar`；`groupId` 為 `com.apusic`；`artifactId` 為 `acls-client`；`version` 為 `1.0.9`，版本號通常在jar文件打包時檔名稱已標明；`packaging` 為 `jar`：

```
mvn install:install-file -Dfile=optional/acls-client-1.0.9.jar -DgroupId=com.apusic -DartifactId=acls-client -Dversion=1.0.9 -Dpackaging=jar
```

在pom文件引入。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>acls-client</artifactId>
  <version>1.0.9</version>
</dependency>
```

連線授權中心。以通過JVM引數啟動為例，更多連線方法請參考《金蝶Apusic許可授權中心用戶手冊》。

- `apusic_acls_enable`：是否開啟授權中心認證，取值為 `true` 或 `false`，為 `true` 則表示開啟授權中心認證。沒有該引數或該引數值為 `false`，都表示沒有開啟授權中心認證
- `apusic_acls_authUrls`：授權中心的地址，可設置多個授權地址，格式為 `ip1:port1,ip2:port2`，如果一個授權地址連結失敗，會輪詢其他的地址；如果開啟授權中心認證，則為必填引數，其中端口為授權中心的https端口
- `apusic_acls_ns`：設置該例項所屬的名稱空間名稱，可選引數；默認值為 `public`，具體的名稱空間可以在授權中心管理控制台-系統管理-授權管理檢視
- `apusic_acls_tenant`：設置該例項所屬的租戶名稱，可選引數

```
java -Dapusic_acls_ns=AA_TEMP -Dapusic_acls_enable=true -Dapusic_acls_authUrls=2.x.x.6:6869 -jar spring-
boot-jsp.jar
```

10.5 容器功能說明

10.5.1 安全管理

介紹安全管理相關配置。

10.5.1.1 HTTPS

10.5.1.1.1 簡介

HTTPS（全稱：Hypertext Transfer Protocol Secure），是由 HTTP 加上 TLS/SSL 協議構建的可進行加密傳輸、身份認證的網路協議，可以有效提高網站安全性。金蝶 Apusic 應用伺服器嵌入式部署支援HTTPS。

10.5.1.1.2 配置方法

使用HTTPS時，可以通過Spring Boot 配置載入對應 `application.properties`、`application.yml` 等文件載入配置項。

`application.properties` 配置如下：

```
#https配置
#應用伺服器監聽端口
server.port=8089
#是否開啟SSL協議
server.ssl.enabled=true
#配置SSL證書目錄
server.ssl.key-store=classpath:keystore.jks
#SSL證書型別
server.ssl.key-store-type=JKS
#私鑰密碼
server.ssl.key-password=[YOUR_PASSWORD]
#金鑰庫文件密碼
server.ssl.key-store-password=[YOUR_PASSWORD]
#伺服器使用的密碼和證書在金鑰庫的別名
server.ssl.key-alias=apusicuserkey
#客戶端證書驗證，值為 need,want,none 之一
server.ssl.client-auth=none
#信任庫文件，用於驗證客戶端證書
server.ssl.trust-store=classpath:truststore.jks
#信任庫密碼
server.ssl.trust-store-password=[YOUR_PASSWORD]
#信任庫型別
server.ssl.trust-store-type=JKS
```

```
#允許使用的密碼套件
```

```
server.ssl.ciphers=*
```

10.5.1.2 國密配置

10.5.1.2.1 簡介

金蝶 Apusic 應用伺服器V10嵌入式部署支援國密SSL協議通訊，支援國家密碼局認定的國產密碼演算法。

10.5.1.2.2 配置方法

註冊國密實現jar包

將 金蝶Apusic應用伺服器V10嵌入式部署 的國密實現 Jar 包 - `${APUSIC_HOME}/optional/aas-sm-3.80.1.jar` 註冊到 Maven 庫中，以備應用打包引用。

```
mvn install:install-file -Dfile=aas-sm-3.80.1.jar -DgroupId=com.apusic -DartifactId=aas-sm -
Dversion=3.80.1 -Dpackaging=jar
```

引入依賴

在 `pom.xml` 文件中新增國密實現依賴。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-sm</artifactId>
  <version>3.80.1</version>
</dependency>
```

配置文件

引入依賴之後，需要在專案的 `application.properties` 或 `application.yml` 新增國密SSL配置。

`application.properties` 配置如下：

```
#國密配置：
#表示SSL通訊協議使用國密通訊協議
server.ssl.protocol=SMv1.1
#國密證書私鑰型別（PKCS12或JKS）
server.ssl.key-store-type=PKCS12
#配置國密金鑰庫文件
server.ssl.key-store=classpath:keystore.p12
#私鑰密碼
server.ssl.key-password=[YOUR_PASSWORD]
#金鑰庫文件密碼
server.ssl.key-store-password=[YOUR_PASSWORD]
#客戶端證書驗證，值為 need,want,none 之一，雙向認證時配置值為need
server.ssl.client-auth=none
#信任庫文件，用於驗證客戶端證書，雙向認證時配置，此信任庫需儲存智慧密碼鑰匙（USB Key）的根證書
server.ssl.trust-store=classpath:truststore.jks
#信任庫密碼，雙向認證時配置
server.ssl.trust-store-password=[YOUR_PASSWORD]
#信任庫型別，雙向認證時配置（PKCS12或JKS）
server.ssl.trust-store-type=JKS
```

啟動應用

Spring Boot 應用中使用國密需要在應用啟動中新增 `-Dcom.apusic.security.ssl.EnableGMTLS=true` 和 `-Djava.security.egd=file:/dev/./urandom` 引數，例如：

```
java -Dcom.apusic.security.ssl.EnableGMTLS=true -Djava.security.egd=file:/dev/./urandom -jar springboot-gm-demo.jar
```

10.5.1.2.3 客戶端配置

專案配置國密SSL協議之後，需要使用支援國密SSL協議的瀏覽器訪問，例如360安全瀏覽器、奇安信等。瀏覽器需要開啟國密通訊功能。

然後需要在客戶端中匯入伺服器證書，例如 `sm2.ca.pem`。

以360安全瀏覽器為例，

1. 進入“設置”-“選項”-“安全設置”，勾選“國密通訊協議”，設置“國密SSL通訊白名單”。
2. “隱私安全設置”-“管理HTTPS/SSL證書...”，“受信任的根證書頒發機構”，匯入根證書。

重啟瀏覽器。

选项

基本设置

界面设置

标签设置

鼠标手势

快捷键

高级设置

实验室

安全设置

核心安全防护

☐ 多层系统防护 文件，驱动，进程，注册表多层防护，保护系统安全

☒ 沙箱防护 页面沙箱防护，防止未知病毒侵害

☐ 挖矿防护 禁止执行数字货币挖矿脚本，防止硬件资源被利用

☒ 隔离防护 网页，代码，插件，CPU进程隔离

☒ 系统级防注入 防恶意代码注入，保护浏览器安全

隐私安全设置

清理上网痕迹设置...

管理HTTPS/SSL证书...

☐ 检查服务器证书吊销状态

☒ 拦截证书风险

☒ 开启安全键盘功能，输入密码时使用虚拟键盘，可防止密码被键盘记录器窃取

☒ 自动停用来源不明的扩展

☐ 开启“禁止跟踪(DNT)”功能

国密通信协议

☒ 启用国密SSL协议支持

国密SSL通讯白名单

HTTPS协议优先走国密SSL协议

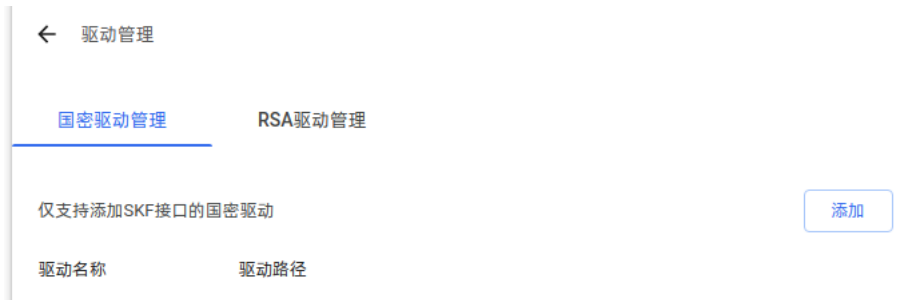
根证书管理

信任的根证书

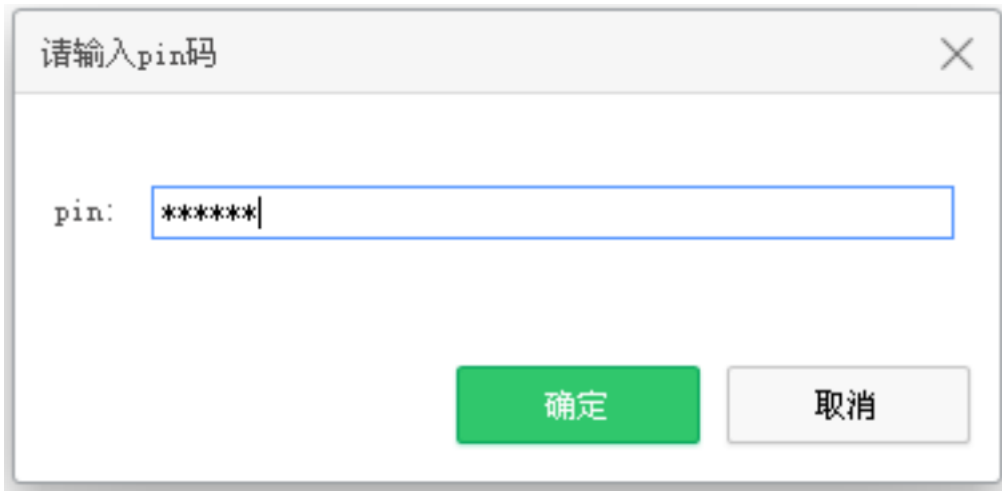
3. 如果是雙向認證則需要設置智慧密碼鑰匙（USB Key）的驅動。

在360安全瀏覽器v13.4為例：

“安全隱私”->“驅動安全”->“驅動管理”->“國密驅動管理” 頁面中點選新增國密驅動。



設置成功後，訪問伺服器，應會出現輸入PIN頁面，輸入正確的PIN碼，即可正確訪問頁面。



10.5.1.2.4 其他問題

如果使用 Oracle JDK，由於 JVM 默認的加密強度限制，需要下載 JCE 無限強度加密策略文件（Unlimited Strength Jurisdiction Policy Files），解壓後用其中的 local_policy.jar 和 US_export_policy.jar 替換 %JAVA_HOME%\jre\lib\security\ 目錄下的同名文件，解除加密強度限制。附件為 JDK 8 的對應文件，其他版本的 JDK 需要至官網下載對應版本文件。

[JCE 文件](#)

10.5.2 國際化支援

10.5.2.1 簡介

為方便不同區域用戶使用金蝶Apusic應用伺服器V10嵌入式部署支援，金蝶 Apusic 應用伺服器V10嵌入式部署提供國際化支援。

10.5.2.2 配置方法

默認日誌輸出採用的是英文格式輸出。如修改成中文格式需要在 spring boot 的配置文件中編寫程式碼定義日誌輸出格式配置，如下所示。

```
@Bean
public Locale locale() {
    return Locale.CHINA;
}
```

10.5.3 日誌管理

10.5.3.1 應用日誌

10.5.3.1.1 簡介

金蝶 Apusic 應用伺服器V10嵌入式部署支援日誌功能，支援整合Spring Boot方法實現。

10.5.3.1.2 使用方法

可以通過Spring Boot 配置載入對應 application.properties 或 application.yml 配置文件，配置Spring Boot 標準日誌配置資訊。

application.properties 配置如下：

```
#日誌配置
#關閉日誌輸出
#logging.level.root: OFF
#日誌級別
logging.level.root=info
#使用相對路徑的方式設置日誌輸出的位置
#logging.file.path=logs
#控制台日誌輸出格式
logging.pattern.console=%d{yyyy-MM-dd hh:mm:ss} [%thread] %-5level %logger{50} - %msg%n
#日誌文件輸出格式
logging.pattern.file=%d{yyyy-MM-dd} === [%thread] === %-5level === %logger{50} === - %msg%n
#定義日誌檔名稱
logging.file.name=logs/aams-log-test.log
```

10.5.3.2 訪問日誌

10.5.3.2.1 簡介

金蝶 Apusic 應用伺服器V10嵌入式部署支援訪問日誌記錄輸出能力，整合Spring Boot方法配置對接。

10.5.3.2.2 使用方法

可以通過Spring Boot 配置載入對應 application.properties 或 application.yml 配置文件，配置日誌資訊。

application.properties 配置如下：

```
server.aas.basedir=./
server.aas.accesslog.enabled=true
```

10.5.4 監控支援

10.5.4.1 簡介

金蝶 Apusic 應用伺服器V10嵌入式部署提供監控功能，支援整合 springboot actuator 框架監控對應資源（需在 pom 中自行引入 spring-boot-starter-actuator 依賴，嵌入式 starter 不自帶該模組）。

10.5.4.2 使用方法

直接引入 aams-spring-boot-starter-x.x.RELEASE.jar

10.5.4.2.1 SPRINGBOOT1.X 的版本

在專案 pom.xml 中引入 aams-spring-boot-starter 的依賴。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter</artifactId>
  <version>1.5.20.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  <version>1.5.20.RELEASE</version>
</dependency>
```

開啟監控

在 spring boot 的配置文件中配置開啟監控，在 application.properties 配置如下。

```
#配置端口
management.port=9998
```

#配置監控訪問上下文

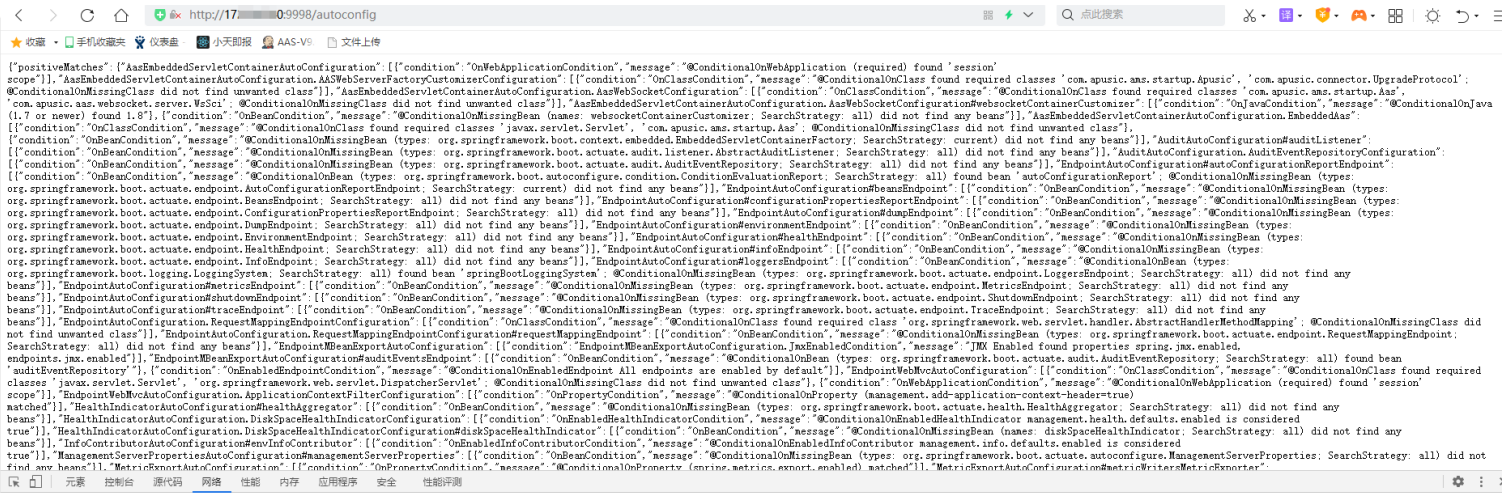
#management.context-path=/autoconfig

#設置是否開啟安全性

management.security.enabled=false

檢視監控

啟動服務，格式為 `http://[IP]:[management.port]/[management.context-path]/[訪問目錄項]`。如訪問 `http://[IP]:9998/autoconfig`。



訪問目錄描述

路徑	說明
/configprops	描述配置屬性如何注入Bean
/beans	描述配置應用程式上下文裡全部Bean以及其關係
/autoconfig	提供自動配置報告，記錄自動配置通過情況
/env	獲取全部環境屬性
/health	報告應用程式的監控指標
/loggers	獲取loggers
/dump	獲取線程活動快照
/metrics	報告各種應用程式度量資訊，比如記憶體用量和HTTP請求計數
/trace	提供基本的http請求跟蹤資訊
/mappings	報告全部URL路徑

10.5.4.2.2 SPRINGBOOT2.X 的版本

在專案 `pom.xml` 中引入 `aams-spring-boot-starter` 的依賴。

如果springboot版本是 `springboot2.x-springboot2.4` 版本，則 `pom` 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter-all</artifactId>
  <version>2.1.7.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
```

```
<version>2.1.7.RELEASE</version>
</dependency>
```

如果springboot版本是 springboot2.4 版本，則 pom 文件中引入如下內容：

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter-all</artifactId>
  <version>2.4.0</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  <version>2.4.0</version>
</dependency>
```

開啟監控

在 spring boot 的配置文件配置開啟監控，在 application.properties 配置如下。

```
#actuator端口 如果不配置做默認使用上面server.port端口
management.server.port=9999
#默認值訪問health,info端點，用*可以包含全部端點
management.endpoints.web.exposure.include=*
#修改訪問路徑
management.endpoints.web.base-path=/actuator
```

檢視監控

啟動服務，格式為 http://[IP]:[management.server.port]/[management.endpoints.web.base-path] 。如訪問 http://[IP]:9999/actuator 。

```
[{"_links":{"self":{"href":"http://172.24.2.40:9999/actuator","templated":false},"beans":{"href":"http://172.24.2.40:9999/actuator/beans","templated":false},"caches-caches":{"href":"http://172.24.2.40:9999/actuator/caches/{cache}","templated":true},"caches":{"href":"http://172.24.2.40:9999/actuator/caches","templated":false},"health-path":{"href":"http://172.24.2.40:9999/actuator/health/{*path}","templated":true},"health":{"href":"http://172.24.2.40:9999/actuator/health","templated":false},"info":{"href":"http://172.24.2.40:9999/actuator/info","templated":false},"conditions":{"href":"http://172.24.2.40:9999/actuator/conditions","templated":false},"configprops":{"href":"http://172.24.2.40:9999/actuator/configprops","templated":false},"env-toMatch":{"href":"http://172.24.2.40:9999/actuator/env/{toMatch}","templated":true},"env":{"href":"http://172.24.2.40:9999/actuator/env","templated":false},"loggers":{"href":"http://172.24.2.40:9999/actuator/loggers","templated":false},"loggers-names":{"href":"http://172.24.2.40:9999/actuator/loggers/{name}","templated":true},"heapdump":{"href":"http://172.24.2.40:9999/actuator/heapdump","templated":false},"threaddump":{"href":"http://172.24.2.40:9999/actuator/threaddump","templated":false},"metrics":{"href":"http://172.24.2.40:9999/actuator/metrics","templated":false},"metrics-requiredMetricName":{"href":"http://172.24.2.40:9999/actuator/metrics/{requiredMetricName}","templated":true},"scheduledtasks":{"href":"http://172.24.2.40:9999/actuator/scheduledtasks","templated":false},"mappings":{"href":"http://172.24.2.40:9999/actuator/mappings","templated":false}}}]
```

訪問目錄描述

路徑	說明
/actuator	actuator服務訪問上下文
/configprops	描述配置屬性如何注入Bean
/beans	描述配置應用程式上下文裡全部Bean以及其關係
/caches	獲取快取資訊
/conditions	提供自動配置，記錄自動配置通過情況
/env	獲取全部環境屬性
/health	報告應用程式的監控指標
/loggers	獲取loggers
/heapdump	獲取heapdump
/metrics	報告各種應用程式度量資訊，比如記憶體用量和HTTP請求計數

/threaddump	獲取threaddump
/httptrace	提供基本的http請求跟蹤資訊
/mappings	報告全部URL路徑
/scheduledtasks	獲取計劃任務

10.6 示例操作

示例 `spring-boot-jsp` 使用AAS作為springboot應用的執行容器。本示例基於 Spring Boot 2.4 開發，需要將 AAS 嵌入式依賴包安裝到本地 Maven 倉庫，才能正常引入並開發應用。

10.6.1 環境準備

- **應用工程**：使用的應用工程
- **AAS嵌入式部署產品包**：示例以 `AAMS-V10-embed.tar.gz` 作為參考
- **maven環境**：安裝有可用的maven環境，用於構建應用
- **Java環境**：準備有可用的JDK，版本需要在JDK 8
- **AAS授權文件**：準備對應的AAS授權文件，或者對接的Apusic授權中心

10.6.2 AAS部署

1. 需要部署AAS嵌入式部署產品包，解壓 `AAMS-V10-embed.tar.gz`。
2. 通過終端進入 `AAMS-V10-embed/starter-2.4` 目錄，輸入以下命令安裝依賴到本地 Maven 倉庫：

```
mvn install:install-file -Dfile=aams-spring-boot-starter-all-2.4.0.jar -DgroupId=com.apusic -DartifactId=aams-spring-boot-starter-all -Dversion=2.4.0 -Dpackaging=jar
```

3. 訪問 maven 的本地倉庫地址" `.../repository/com/apusic/aams-spring-boot-starter-all/2.4.0/` "，出現該目錄及 `.jar` 文件，即表示依賴已成功安裝到本地 Maven 倉庫，可在專案中正常引用。

10.6.3 應用工程

建立或匯入應用工程，示例 `spring-boot-jsp` 原始碼如下：

I-demo\spring-boot-jsp	
名称	^
 src	
 .gitignore	
 pom.xml	

10.6.4 pom文件

pom文件如下：

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.apusic.test</groupId>
  <artifactId>spring-boot-jsp</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <properties>
    <failOnMissingWebXml>false</failOnMissingWebXml>
```

```

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <aams.version>10.0.0</aams.version>
</properties>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>2.4.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <version>2.4.0</version>
    <exclusions>
      <exclusion>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
    <version>2.4.0</version>
  </dependency>
  <dependency>
    <groupId>com.apusic</groupId>
    <artifactId>aams-spring-boot-starter-all</artifactId>
    <version>2.4.0</version>
  </dependency>
  <dependency>
    <groupId>com.apusic</groupId>
    <artifactId>aas-sm</artifactId>
    <version>3.80.1</version>
  </dependency>
  <dependency>
    <groupId>com.apusic</groupId>
    <artifactId>aas-embed-jasper</artifactId>
    <version>${aams.version}</version>
  </dependency>
  <dependency>
    <groupId>com.apusic</groupId>
    <artifactId>aas-embed-el</artifactId>
    <version>${aams.version}</version>
  </dependency>
  <dependency>

```

```

<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<version>2.4.0</version>
<exclusions>
  <exclusion>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
  </exclusion>
</exclusions>
</dependency>
<dependency>
  <groupId>javax.validation</groupId>
  <artifactId>validation-api</artifactId>
  <version>2.0.1.Final</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.10.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>2.4.0</version>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
  <resources>
    <resource>
      <directory>src/main/java</directory>
      <includes>
        <include>/**/*.properties</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/resources</directory>

```

```

    </resource>
    <resource>
      <directory>src/main/webapp</directory>
      <targetPath>META-INF/resources</targetPath>
      <includes>
        <include>**/*</include>
      </includes>
    </resource>
  </resources>
</build>
</project>

```

10.6.5 構建應用包

進入工程目錄下，使用maven構建，執行 `mvn package`。

10.6.6 執行應用

執行前需要將授權文件 `license.xml` 放置與應用 `spring-boot-jsp.jar` 同一目錄，執行：

```
java -jar spring-boot-jsp.jar
```

10.6.7 訪問應用

客戶端訪問<http://IP:PORT/MyServlet>，如<http://localhost:8080/MyServlet>。

10.6.8 以除錯模式執行

如果需要啟動除錯模式（debug），執行：

```
java -agentlib:jdwp=transport=dt_socket,address=8000,server=y,suspend=n -jar spring-boot-jsp.jar
```

10.6.9 停止應用

根據應用啟動方式，可選擇對應的停止操作：

- 前臺啟動停止

如果應用以前臺方式啟動（直接在終端執行，未後臺掛起），**直接關閉或退出當前終端**，即可終止應用程序，完成停止操作。

- 後臺啟動停止

如果應用以後臺方式啟動（例如通過 `nohup`、`&` 等方式執行），需按以下步驟操作：

1. 查詢程序號

執行以下任一命令，獲取應用程序 ID（PID）：

```

# 方式1：通過程序關鍵詞查詢
ps -ef | grep apusic

# 方式2：通過JVM程序工具查詢
jps -lvm

```

2. 停止程序

到程序號後，執行 `kill` 命令終止程序：

```

# 先嚐試優雅停止
kill [程序號]

# 若程序無法優雅停止，可強制終止
kill -9 [程序號]

```

11 鏈路追蹤與調用鏈分析

應用伺服器支援整合鏈路追蹤工具。SkyWalking 和 Pinpoint 都是分散式系統的鏈路追蹤工具，用於幫助開發人員理解和優化微服務架構中的系統性能和調用關係。

本章內容對獨立部署與嵌入式部署（Spring Boot 整合）均適用。

11.1 與 Pinpoint 整合

11.1.1 簡介

Pinpoint 是一款開源的 Java 應用性能監控（APM）工具，主要用於分散式系統的性能監控和故障排查，能夠對 Java 微服務架構進行全方位的監控和分析。它通過位元組碼增強技術，在零程式碼侵入的前提下，自動採集應用中的鏈路追蹤資料、性能指標資料、線程堆疊資料等，幫助開發者和運維人員快速瞭解系統的執行狀態，定位性能瓶頸和故障點。

11.1.2 相關資源

- 金蝶Apusic應用伺服器產品包
- 金蝶Apusic應用伺服器pinpoint外掛，`aas-pinpoint-[version].jar`（位於產品安裝目錄 `plugins/pinpoint/` 下，如 `aas-pinpoint-2.5.3.jar`、`aas-pinpoint-2.5.4.jar`）
- pinpoint v2.5.3及以上
- pinpoint-agent v2.5.3及以上

11.1.3 Pinpoint-agent安裝

下載並安裝pinpoint-agent，建議使用v2.5.3及以上版本。安裝包及安裝說明參考官網介紹，在此不作詳細說明。

pinpoint-agent需要與金蝶Apusic應用伺服器安裝在同一伺服器上。

11.1.4 應用伺服器APM外掛配置

金蝶Apusic應用伺服器的pinpoint外掛位於產品安裝目錄 `plugins/pinpoint/` 下。將應用伺服器pinpoint外掛，如 `aas-pinpoint-2.5.3.jar`，複製至pinpoint-agent對應目錄下，如 `pinpoint-agent-2.5.3/plugin/`。

配置pinpoint-agent，如 `pinpoint-agent-2.5.3\profiles\release\pinpoint.config` 修改如下配置：

```
profiler.collector.ip=127.0.0.1 #收集器的IP地址
profiler.transport.grpc.collector.ip=127.0.0.1 # 要用於指定在使用 gRPC (Google Remote Procedure Call) 作為傳輸協議進行性能資料傳輸時，收集器 (Collector) 的 IP 地址
#下面兩項根據自己需求修改，如果不改，可能會採集不到監控全部資料
profiler.sampling.counting.sampling-rate=1 #計數形式設置取樣率，取樣率一般是一個介於 0 和 1 之間的小數。例如，0.1 表示在所有可能的事件中，選擇 10% 的事件進行取樣；0.01 則表示選擇 1% 的事件進行取樣。訪問量不大的情況，建議全部採集
profiler.sampling.percent.sampling-rate=100 #百分比形式設置取樣率
```

11.1.5 應用伺服器配置

應用伺服器整合pinpoint-agent。編輯啟動指令碼，如 `apusic.sh`，在 `JAVA_OPTS` 新增pinpoint-agent資訊

```
JAVA_OPTS="$JAVA_OPTS -javaagent:/opt/pinpoint/pinpoint-agent-2.5.3/pinpoint-bootstrap-2.5.3.jar -Dpinpoint.agentId=test -Dpinpoint.applicationName=AASV10"
```

- `-javaagent`：pinpoint-agent所在的位置，建議填寫實際目錄，該目錄需要開放許可權
- `-Dpinpoint.agentId`：設置pinpoint-agent的ID，與被監控的其他專案不能相同
- `-Dpinpoint.applicationName`：採集專案的名稱

11.1.6 啟動應用伺服器

配置完成後，進入應用伺服器目錄 `${APUSIC_HOME}/bin` 執行啟動命令，如 `apusic.sh run`，啟動金蝶Apusic應用伺服器。啟動時將會同時啟動pinpoint-agent。

11.1.7 嵌入式部署配置

如果是嵌入式部署方式，參考以下方法啟用：

```
java -javaagent:/opt/pinpoint/pinpoint-agent-2.5.3/pinpoint-bootstrap-2.5.3.jar -Dpinpoint.agentId=test -Dpinpoint.applicationName=test -jar test.jar
```

11.1.8 Pinpoint APM操作

Pinpoint（APM監控工具）安裝需部署 Collector、Web 和 Agent 三部分，產品包及安裝步驟可檢視官網，在此不作詳細介紹。

安裝完成後啟動HBase、collector 服務、web 服務。

訪問 Pinpoint Web 控制台，默認為 `http://IP:8080`。

檢視應用列表是否出現已接入的應用。

檢視調用鏈、性能指標是否正常上報。

11.2 與 SkyWalking 整合

11.2.1 簡介

SkyWalking 是一款開源的應用性能監控（APM）工具，主要用於分散式系統的性能監控和故障排查，能夠對微服務架構、雲原生應用等進行全方位的監控和分析。它通過收集和分析應用系統中的各種資料，如鏈路追蹤資料、性能指標資料等，幫助開發者和運維人員快速瞭解系統的執行狀態，定位性能問題和故障點。

11.2.2 相關資源

- 金蝶Apusic應用伺服器產品包
- 金蝶Apusic應用伺服器skywalking外掛，`aas-skywalking-[version].jar`（位於產品安裝目錄 `plugins/skywalking/` 下，如 `aas-skywalking-8.3.0.jar`）
- SkyWalking APM v9.6.0及以上
- Java Agent for Apache SkyWalking v9.0.0及以上

11.2.3 Java Agent for Apache SkyWalking安裝

下載並安裝Java Agent for Apache SkyWalking，建議使用v9.0.0及以上版本。安裝包及安裝說明參考官網介紹，在此不作詳細說明。

Java Agent for Apache SkyWalking需要與金蝶Apusic應用伺服器安裝在同一伺服器上。

11.2.4 應用伺服器APM外掛配置

金蝶Apusic應用伺服器的skywalking外掛位於產品安裝目錄 `plugins/skywalking/` 下，例如 `aas-skywalking-8.3.0.jar`。

將 `aas-skywalking-8.3.0.jar` 放置Agent的plugins目錄中，如 `/opt/skywalking/skywalking-agent/plugins/`。

如果應用伺服器和SkyWalking Agent與SkyWalking APM是遠端通訊，需要設置SkyWalking Agent的配置文件 `agent.config` 中的 `collector.backend_service` 為實際的SkyWalking APM的IP和端口。

11.2.5 應用伺服器配置

應用伺服器整合skywalking-agent。編輯啟動指令碼，如 `apusic.sh`，在 `JAVA_OPTS` 新增skywalking-agent資訊，示例如下：

```
JAVA_OPTS="$JAVA_OPTS -javaagent:/opt/skywalking/test/agent/skywalking-agent/skywalking-agent.jar -
DSW_AGENT_COLLECTOR_BACKEND_SERVICES=127.0.0.1:11800 -DSW_AGENT_NAME=AASV10"
```

- `-javaagent`：skywalking-agent所在的位置，建議填寫實際路徑，該路徑需要開放許可權
- `-DSW_AGENT_COLLECTOR_BACKEND_SERVICES`：skywalking的服務端ip和資料採集端，應根據實際填寫
- `-DSW_AGENT_NAME`：定義註冊的服務名稱

11.2.6 啟動應用伺服器

配置完成後，進入應用伺服器目錄 `${APUSIC_HOME}/bin` 執行啟動命令，如 `apusic.sh run`，啟動金蝶Apusic應用伺服器。啟動時將會同時啟動Java Agent for Apache SkyWalking。

11.2.7 嵌入式部署配置

如果是嵌入式部署方式，參考以下方法啟用：

```
java -javaagent:/opt/skywalking/test/agent/skywalking-agent/skywalking-agent.jar -
DSW_AGENT_COLLECTOR_BACKEND_SERVICES=127.0.0.1:11800 -DSW_AGENT_NAME=test -jar test.jar
```

11.2.8 Skywalking APM操作

安裝SkyWalking APM，產品包及安裝步驟可檢視官網，在此不作詳細介紹。

啟動SkyWalking APM，通常啟動方式為，進入SkyWalking APM安裝目錄的bin目錄，執行 `./startup.sh`。啟動方式可檢視官網，在此不作詳細介紹。

啟動前需要注意JDK安裝環境，需要安裝對應版本，例如SkyWalking APM v9.6.0使用JDK11。

訪問SkyWalking APM的Web管控平台，通常訪問方式為<http://IP:8080>。詳細配置及使用方式可檢視官網，在此不作詳細介紹。

可檢視服務，例如選擇AAS-V10。

11.2.9 問題定位

1. 如果應用伺服器啟動失敗，檢視應用伺服器的日誌，如 `server.log`。
2. 如果Java Agent for Apache SkyWalking啟動失敗，進入Agent的安裝目錄對應的日誌位置，如 `/opt/skywalking/skywalking-agent/logs/`，檢視skywalking-agent的日誌。
3. 如果是SkyWalking APM啟動失敗，進入SkyWalking APM安裝目錄，如 `/opt/skywalking/apache-skywalking-apm-bin/logs/`，檢視APM的日誌。

11.3 調用鏈分析

應用伺服器自帶調用鏈分析工具，位於安裝目錄 `{APUSIC_HOME}/bin/apm` 下，無需另行下載安裝。

11.3.1 配置說明

配置文件為 `{APUSIC_HOME}/bin/apm/conf.properties`。

配置引數	說明	預設值
cost	指定記錄的最低請求耗時，即超過該值的請求才會被記錄，單位為毫秒	3000
methodMinCost	指定最小方法耗時，即超過該值的方法調用才會被記錄，單位為毫秒	200
logDir	設置跟蹤結果的儲存目錄，該目錄需要設置讀寫許可權	sr
incPkgs	指定需要跟蹤的類或包名，多個類或包以逗號分隔	
excPkgs	指定排除追蹤的類或包名，多個類或包以逗號分隔。如果應用的類比較多，應該儘量將不關心的包排除，以免影響性能	com.apusic.,org.,com.google.,com.thoughtworks.,net.sf.,com.ibm.,com.mysql.,oracle.,com.fasterxml.,com.netflix.eureka.,io.micrometer.core.instrument.binder.logging.
batchRetransform	是否批次轉換類，一旦有個類失敗就會整個失敗，為false表示一個類一個類轉換，其中一個失敗了不會影響別的類	true
outputClass	是否輸出轉換後的類，輸出的類在apm根目錄的classes下	false
verifyClass	是否驗證轉換後的類	false

11.3.2 操作方法

- 1、檢視相關程序，如http監聽服務6888端口的程序。

```
netstat -nltp |grep 6888
```

- 2、進入調用鏈分析工具目錄，啟動調用鏈分析工具。`${PID}` 為具體的程序號。

```
java -jar client.jar pid=${PID}
```

- 3、接上一步，在aapm命令列頁面執行 `trace start` 開啟trace。

```
trace start
```

```
[root@linux-4-180 apm]# java -jar client.jar pid=9137
JAVA_HOME: /java/jdk8u462-b08/jre
Try to attach process 9137
Start args: [-jar, /opt/qiumailun/aasv10.0.8/slowsql/ApusicAS/tools/monitor/apm/core.jar, pid=9137, host=localhost, port=5888]
Found java: /java/jdk8u462-b08/jre/bin/java
Found java: /java/jdk8u462-b08/jre/../bin/java
Found tools.jar: /java/jdk8u462-b08/jre/../lib/tools.jar
Picked up JAVA_TOOL_OPTIONS:
Attach process 9137 success.
Connected to localhost:5888 successfully.
[aapm] trace start
trace started successfully, transformed 213 classes, failed 0 classes, cost 16670ms.
```

4、訪問應用，此時在 `${APUSIC_HOME}/bin/apm/sr` 目錄自動生成 調用鏈分析資料 .xml 文件。

5、執行分析命令，生成轉換後的html文件。 `${APUSIC_HOME}` 為應用伺服器安裝目錄， `[tr_name]` 為實際的調用鏈分析資料檔名。

```
./flame.sh ${APUSIC_HOME}/bin/apm/sr/[tr_name].xml
```

```
[root@linux-4-180 apm]# ./flame.sh /opt/.../ApusicAS/tools/monitor/apm/sr/trace-2026-05-09-15-22-20.xml
Flame graph generated: /opt/.../ApusicAS/tools/monitor/apm/sr/trace-2026-05-09-15-22-20.html
[root@linux-4-180 apm]#
```

6、下載HTML文件，檢視調用鏈情況。

12 附錄

12.1 附錄A listeners配置說明

配置引數	說明	類名
aprlifecycleListener	使用 APR 進行接收客戶端請求，在應用伺服器初始化之前、初始化之前嘗試初始化 APR 庫，成功則使用 APR 接受處理客戶端請求，應用伺服器銷燬之後，該監聽器會做 APR 的清理工作	com.apusic.ams.core.AprLifecycleListener
jreMemoryLeakPreventionListener	防止 JRE 記憶體洩露	com.apusic.ams.core.JreMemoryLeakPreventionListener
globalResourcesLifecycleListener	初始化JNDI 資源的 MBean	com.apusic.ams.mbeans.GlobalResourcesLifecycleListener
threadLocalLeakPreventionListener	防止 ThreadLocal 對象帶來的記憶體洩漏	com.apusic.ams.core.ThreadLocalLeakPreventionListener

12.2 附錄B resources配置說明

配置引數	說明	值型別	默認值
name	資源名稱, 必須唯一。如"userDatabase"表示配置用戶許可權	字元	
auth	授權範圍	字元	Container
type	型別	字元	com.apusic.ams.UserDatabase
description	描述	字元	User database that can be updated and saved
factory	資源工廠	字元	com.apusic.ams.users.MemoryUserDatabaseFactory
pathname	路徑名稱	字元	conf/aas-users.xml

12.3 附錄C executors 配置說明

配置引數	說明	值型別	默認值
name	線程池名稱, 供配置文件其他地方引用，必須唯一	字元	
namePrefix	線程池裡面的線程的字首	字元	
maxThreads	最大線程數	整型	200
minSpareThreads	最小空閒線程數	整型	25
maxIdleTime	當線程數大於 minSpareThreads 時，空閒線程銷燬前最大空閒時間，單位毫秒	整型	60000
maxQueueSize	線程池佇列最大任務數，超過後將被拒絕	整型	
prestartminSpareThreads	是否預先啟動最小空閒線程	布林	true

12.4 附錄D endpoints配置說明

配置引數	說明	值型別	默認值
port	監聽端口	整型	
protocol	處理協議，可選的有 HTTP/1.1、AJP/1.3	字元	HTTP/1.1
connectionTimeout	連線超時時間，讀取 HTTP 請求行超時時間。單位毫秒	數字	30000
redirectPort	如果是 https，重定向端口	整型	

executor	使用的線程池	字元	
maxConnections	最大併發數	整型	10000
enableLookups	是否啟用 DNS 查詢，如果設置 true，則 request.getRemoteHost() 時，將調用 DNS 查詢，否則只返回 IP 值	布林值	false
acceptCount	backlog 大小，達到 maxConnections 後，最大佇列數，超出後，請求將被拒絕	整型	100
maxPostSize	FORM URL 方式的最大提交資料。單位位元組	數字	104857600
maxHttpHeaderSize	請求或響應的最大 http header 大小。單位位元組	數字	8192 (8KB)
compression	是否啟用壓縮，可選值為：off,on,force	字元	on
compressionMinSize	壓縮文件閾值，小於該值不會進行壓縮	整型	2048
compressibleMimeType	逗號隔開的可壓縮的 MIME 型別	字元	text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image/png,image/gif,audio/mpeg,video/mp4
connectionUploadTimeout	資料上傳超時時間。僅當 disableUploadTimeout=false 時生效。單位毫秒	數字	
disableUploadTimeout	是否停用上傳超時	布林值	true
URIEncoding	配置 url 中引數編碼	字元	UTF-8
processorCache	快取 Processor 的最大數量	整型	20000
tcpNoDelay	是否停用了 Nagle 演算法，允許小包的傳送	布林值	true
keepAliveTimeout	keepalive 的超時時間。單位毫秒	數字	15000
maxKeepAliveRequests	保持 keepalive 的連線的個數，超過就會關閉連線。-1 不限制	整型	100
address	伺服器監聽地址	IP地址	0.0.0.0 或 ::
maxCookieCount	最大 Cookie 數量	整型	200
relaxedPathChars	允許請求行中的特殊字元	字元	
relaxedQueryChars	允許請求行中引數的特殊字元	字元	
useAsyncIO	是否使用非同步 IO API	布林值	true
useSendfile	是否使用 sendfile 功能。使用 sendfile 後將不壓縮響應	布林值	true
SSLEnabled	是否允許 SSL 通訊	布林值	false
bindOnInit	是否在伺服器初始化階段繫結端口，設置為 false 時延遲到首次請求時繫結（優雅停機由 servers 模組的 gracefulStopAwaitMillis 控制，與本引數無關）	布林值	true
enabledLongThread	是否開啟慢線程監控	布林值	false
longThreadLogPath	指定慢線程日誌輸出的目錄位置	字元	
longThreadThreshold	定義慢線程閾值，超過該閾值為慢線程，單位為毫秒	字元	60000
longThreadCycleTime	設置慢線程檢測週期，單位為毫秒	字元	3000
longThreadRollingInterval	設置慢線程日誌滾動時間間隔，單位為毫秒	字元	300000

12.4.1 SSLHostConfig配置引數

配置引數	說明	值型別	默認值
protocols	允許與客戶端通訊的協議名字。可以是以下的組合:SSLv2Hello/SSLv3/TLSv1/TLSv1.1/TLSv1.2/TLSv1.3/SMv1.1/all。出廠安全配置默認僅啟用 TLSv1.2+TLSv1.3	字元	all

sslProtocol	伺服器使用的 SSL 協議，用於建立 SSLContext	字元	TLS
truststoreFile	信任庫文件，用於驗證客戶端證書	字元	系統屬性 javax.net.ssl.trustStore
truststoreType	信任庫型別	字元	JKS
truststorePassword	信任庫密碼	字元	系統屬性 javax.net.ssl.trustStorePasswo rd
certificateVerification	證書驗證，如果設置為： required: 客戶端必須返回有效證書。 optional: 希望客戶端返回證書，若客戶端不發證書，也不報錯。 none 不需要客戶端證書，除非客戶端請求 "CLIENT-CERT" 限制的資源	字元	none
ciphers	允許使用的密碼套件	字元	

12.4.2 Certificate配置引數

配置引數	說明	值型別	默認值
certificateFile	PEM 編碼的證書文件	字元	
certificateChainFile	PEM 編碼的證書鏈文件	字元	
certificateKeyFile	PEM 編碼的金鑰文件	字元	
certificateKeyPassword	金鑰文件密碼	字元	
certificateKeyAlias	伺服器使用的密碼和證書在金鑰庫的別名	字元	
certificateKeystoreFile	金鑰庫文件	字元	
certificateKeystorePassword	金鑰庫文件密碼	字元	
certificateKeystoreType	金鑰庫文件型別	字元	JKS
type	證書型別，用於確認密碼套件和證書是匹配的，必須是為 UNDEFINED,RSA,DSS 和 EC 之一。如果 SSLHostConfig 只有一個 Certificate 元素,type 屬性不是必須的。若有多個 Certificate 元素，則必須 指定 type 屬性	字元	UNDEFINED

12.5 附錄E servers 配置說明

配置引數	說明	值型別	默認值
name	服務名稱，必須全域性唯一	字元	
defaultHost	默認虛擬主機名稱	字元	
jvmRoute	在負載均衡使用場景中，Session 粘滯時使用的標籤，在叢集節點中必須唯一。此標籤將字尾在 Session ID 中，此後同一會話的請求將被同一AAS例項處理。	字元	

12.5.1 Realm

配置引數	說明	值型別	默認值
className	Java 類名，此類必須實現 com.apusic.ams.Realm	字元	
dataSourceName	JDBC 數據源的 JNDI 名字	字元	
userTable	用戶表名，表中必須有 userNameCol 和 userCredCol 屬性指定的列	字元	
userNameCol	用戶名稱列	字元	
userCredCol	用戶憑據列（例：密碼）	字元	
userRoleTable	用戶角色表，表中必須有 userNameCol 和 roleNameCol 屬性指定的列	字元	
roleNameCol	角色列	字元	
resourceName	UserDatabase 資源的名字	字元	UserDatabase

cacheSize	快取驗證失敗用戶數	整型	1000
failureCount	鎖定前驗證失敗次數	整型	5
lockOutTime	用戶鎖定時間，在鎖定時間內，驗證失敗將重置計時器，驗證成功不起效。 單位為秒	整型	300

12.5.2 host

配置引數	說明	值型別	默認值
name	虛擬名稱，必須全域性唯一，通常為在 DNS 註冊的網路名	字元	
appBase	應用部署基路徑	字元	webapps
unpackWARs	是否解壓 war 進行部署	布林值	true
autoDeploy	是否自動部署，如果為 true，AAS 將週期性的檢查 appBase 目錄，部署新發現的應用，重新部署更新過的應用	布林值	true
deployOnStartup	啟動時是否自動部署webapps下應用。僅在啟動階段掃描部署一次	布林值	true

12.5.3 application

配置引數	說明	值型別	默認值
path	應用上下文，訪問路徑。同一虛擬主機裡，必須唯一。如果 docBase 在 host 的 appBase 內，則此屬性必須不能設置，不然同一應用可能會被部署兩次。	字元	
docBase	應用的根目錄（應用以目錄形式打包），或者應用（WAR 包）的路徑，可以是絕對路徑或者相對於 host 元素的 appBase 屬性的相對路徑	字元	
useHttpOnly	設置為 true 表示防止客戶端指令碼讀取 JSESSIONID，防止 CSRF/XSS 惡意攻擊	布林值	true
cookie	是否允許 cookies 傳遞 Session ID	布林值	true
crossContext	若設置為 true，則 ServletContext.getContext() 能獲取同一虛擬主機下其他應用的 ServletContext。若設置為 false，則 ServletContext.getContext() 返回 null。	布林值	false
logEffectiveWebXml	若為 true，伺服器在應用啟動時輸出生效的 Web.xml 日誌（INFO）等級，生效的 web.xml 包括默認 web.xml，應用提供的 web.xml、web-fragment.xml 和 annotations 解析結果	布林值	false
override	若設置為 true，則忽略所有全域性和 host 的默認 context 的設置	布林值	true
reloadable	如果設置為 true，則監控 /WEB-INF/classes/ 和 /WEB-INF/lib/，若文件有變化，則重新部署應用	布林值	true
sessionCookieDomain	設置此應用中 Session cookies 的 domain 值	字元	
sessionCookieName	設置 Session cookies 的名字	字元	JSESSIONID
swallowOutput	若設置為 true，應用中輸出到 System.out 和 System.err 的內容將被重定向到應用的 logger 中	布林值	false

12.5.4 servers

配置引數	說明	值型別	默認值
name	定義的server名稱，全域性唯一	字元	
services	引用的服務，僅可新增一個服務	字元	
endpoints	引用的網路和協議配置，多個用逗號隔開	字元	
executors	引用的線程池，多個用逗號隔開，注意：該 server 裡面的 endpoints 裡面引用的線程池也必須在這裡引用，如上面的 http-thread-pool	字元	

gracefulStopAwaitMillis	AAS停止後，等待正在進行中的客戶端請求處理完畢的時長。單位毫秒，為0時表示關閉此功能	字元	
-------------------------	---	----	--

12.6 附錄F containers 配置說明

配置引數	說明	值型別	默認值
name	container名稱，全域性唯一	字元	
port	偵聽端口，用於關閉應用伺服器，不使用可設置為 -1，建議不使用	整型	
servers	該應用伺服器例項引用的伺服器，多個可用逗號隔開，建議使用單個	字元	
resources	該應用伺服器例項引用的資源，比如用戶名稱密碼等，多個可用逗號隔開	字元	
listeners	該應用伺服器例項引用的監聽器，多個可用逗號隔開。引用的是文件開頭配置的監聽器	字元	

12.7 附錄G Resource數據源配置說明

配置引數	說明	值型別	默認值
name	數據源 jndi 名稱	字元	
auth	容器負責資源的連線	字元	
type	繫結型別	字元	
url	資料庫連線 url	字元	
username	用戶名稱	字元	
password	密碼	字元	
maxTotal	最大連線數	字元	
maxIdle	最大空閒連線數	字元	
maxWaitMillis	獲取連線的最大等待時間（單位毫秒），-1 表示無限等待	字元	-1
minIdle	最小空閒連線數	字元	
validationQuery	連線有效性檢測語句	字元	
validationQueryTimeout	檢測語句超時時間，-1表示不超時	字元	-1
testOnBorrow	true 或 false，從池中拿連線的時，是否檢測連線有效性	布林值	true
testOnReturn	true 或 false，還回連線到池裡面時，是否檢測連線有效性	布林值	false
removeAbandonedOnBorrow	true 或 false，是否移除失效的連線	布林值	false
logAbandoned	true 或 false，是否列印移除失效連線的資訊及洩漏程式碼堆疊	布林值	false
testWhileIdle	true 或 false，連線空閒時，是否檢測連線有效性	布林值	false
minEvictableIdleTimeMillis	連線空閒時間，超過該時間會從池中移除（單位為毫秒）	整型	
timeBetweenEvictionRunsMillis	每隔多少時間去檢測一次空閒連線是否超時，默認不檢測（單位為毫秒）	整型	
factory	當設置為“com.apusic.aas.jdbc.pool.DataSourceFactory”時，用於在JNDI數據源配置中，指定由Apusic自帶的連線池實現來建立 javax.sql.DataSource、javax.sql.XADataSource例項，而非第三方或容器默認連線池。 如若設置了factory="com.apusic.aas.jdbc.pool.DataSourceFactory"，引數 maxTotal、maxWaitMillis、removeAbandonedOnBorrow設置無效，需要對應設置maxActive、maxWait、removeAbandoned	字元	
removeAbandoned	true 或 false，是否開啟自動回收洩漏連線。通常用於設置了 factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	布林值	false

removeAbandonedTimeout	連線超時時間，超過該值未歸還視為洩漏，通常`removeAbandoned=true`時才生效	整型	
threshold	定義超過該時間判定為耗時SQL，單位為毫秒（ms）	整型	
notifyPool	true 或 false，控制慢 SQL 觸發後，是否通知連線池核心做狀態聯動。為“true”時，慢查詢觸發時，通知連線池、標記連線、可參與池監控 / 回收邏輯；為“false”時，只打印慢查詢日誌，不通知連線池，無額外池邏輯開銷	布林值	false
logFailed	true 或 false，是否列印錯誤日誌	布林值	false
maxActive	最大活躍連線數，連線池中同一時刻能同時分配給業務使用的最大數據庫連線數量。通常用於設置了 factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	整型	
maxWait	最大等待時間，當連線池已滿（達到 maxActive），新業務請求從連線池獲取空閒連線時，最長阻塞等待的時間，單位毫秒（ms）。通常用於設置了 factory="com.apusic.aas.jdbc.pool.DataSourceFactory"時	整型	

12.8 附錄H Web通用配置

配置引數	說明	值型別
server.port	應用伺服器監聽端口	8080
server.address	應用伺服器監聽地址	
server.servlet-path	用於設置應用程式中所有 Servlet 的基礎路徑。客戶端請求時，請求的 URL 需要加上這個基礎路徑才能訪問到對應的 Servlet。通過設置這個屬性，可以為應用程式的 Servlet 訪問設置一個統一的字首	
server.use-forward-headers	是否在http請求頭中應用X-Forwarded-*	
server.server-header	設置 HTTP 響應頭 Server 的值	
server.max-http-header-size	設置最大 HTTP 訊息頭大小	8192
server.connection-timeout	連線超時時間，讀取 HTTP 請求行超時時間	20000ms
server.session-timeout	會話超時時間，通常用於springboot1.x版本中；springboot2.x及以上版本建議配置server.servlet.session.timeout	
server.error.include-binding-errors	配置在錯誤響應裡是否包括資料繫結錯誤資訊，可設項為： always:無論在什麼情況下，錯誤響應中都會包含資料繫結錯誤資訊。 never:錯誤響應中永遠不會包含資料繫結錯誤資訊。 on-param:只有當請求中包含 trace 引數時，錯誤響應才會包含資料繫結錯誤資訊	never
server.error.include-exception	控制在錯誤響應中是否包含異常類的完整名稱。可設項為"true"或"false"	FALSE
server.error.include-message	用於控制在錯誤響應中是否包含異常訊息的配置屬性。可設項為： always:無論在什麼情況下，錯誤響應中都會包含異常訊息。 never:錯誤響應中永遠不會包含異常訊息。 on-param: 只有當請求中包含 trace 引數時，錯誤響應才會包含異常訊息	never
server.error.include-stacktrace	用於控制在錯誤響應裡是否包含異常堆疊跟蹤資訊。可設項為： always:無論在什麼情況下，錯誤響應中都會包含異常堆疊跟蹤資訊。 never:錯誤響應中永遠不會包含異常堆疊跟蹤資訊。 on-param: 只有當請求中包含 trace 引數時，錯誤響應才會包含異常堆疊跟蹤資訊	never
server.error.path	配置錯誤處理路徑	/error
server.error.whitelabel.enabled	當伺服器發生錯誤時，是否啟用瀏覽器顯示的默認錯誤頁面	TRUE

server.ssl.enabled	是否允許 SSL 通訊	true
server.ssl.ciphers	允許使用的密碼套件	
server.ssl.client-auth	客戶端證書驗證，值為 need,want,none 之一	
server.ssl.enabled-protocols	允許與客戶端通訊的協議名字	
server.ssl.key-alias	伺服器使用的密碼和證書在金鑰庫的別名	
server.ssl.key-password	金鑰庫金鑰密碼	
server.ssl.key-store	金鑰庫文件	
server.ssl.key-store-password	金鑰庫文件密碼	
server.ssl.key-store-provider	金鑰儲存文件的提供者	
server.ssl.key-store-type	金鑰庫文件型別	
server.ssl.protocol	伺服器使用的 SSL 協議，用於建立 SSLContext	TLS
server.ssl.trust-store	信任庫文件，用於驗證客戶端證書	
server.ssl.trust-store-password	信任庫密碼	
server.ssl.trust-store-provider	信任儲存的提供者	
server.ssl.trust-store-type	信任庫型別	
server.servlet.context-parameters.*	Servlet上下文初始化引數	
server.servlet.context-path	應用上下文	
server.servlet.application-display-name	應用顯示名稱	application
server.servlet.jsp.class-name	用於 JSP 的 Servlet 的類名	
server.servlet.jsp.init-parameters	用於配置 JSP Servlet 的 Init 引數	
server.servlet.jsp.registered	JSP Servlet 是否已註冊	
server.servlet.session.store-dir	Session 儲存路徑（用於儲存會話資料的目錄）	
server.servlet.session.timeout	Session 失效時間	30m（30分鐘）
server.servlet.session.tracking-modes	配置會話跟蹤模式。可設項為:COOKIE、URL、SSL	COOKIE
server.servlet.session.persistent	是否在重啟之間保留會話資料,開啟後（設為 true）, AAS會在關閉時將記憶體中的 Session 序列化到文件，重啟時再載入回來，實現會話不去失；關閉（false）則 Session 完全存放在記憶體中，重啟後丟失	FALSE
server.servlet.session.cookie.domain	設置該 Cookie 的域名	
server.servlet.session.cookie.http-only	設置為 true 表示防止客戶端指令碼讀取 JSESSIONID	true
server.servlet.session.cookie.max-age	Cookie 失效的時間	
server.servlet.session.cookie.name	設置 Cookie 的名稱	JSESSIONID
server.servlet.session.cookie.path	Cookie 的使用路徑	
server.servlet.session.cookie.comment	cookie的註釋	
server.servlet.session.cookie.secure	是否始終將cookie標記為安全	
server.servlet.register-default-servlet	是否向容器註冊默認Servlet	
server.aas.resource.allow-caching	是否快取靜態文件	true
server.aas.resource.cache-ttl	快取的 TTL	
server.aas.internal-proxies	匹配可信代理的正規表示式	10\.\d{1,3}\.\d{1,3}\.\d{1,3}192\.\d{1,3}\.\d{1,3}\.\d{1,3}169\.\d{1,3}\.\d{1,3}\.\d{1,3}127\.\d{1,3}\.\d{1,3}\.\d{1,3}172\.\d{1,3}\.\d{1,3}\.\d{1,3}

		\\d{1,3}\\d{1,3} 172\\2[0-9]{1}\\d{1,3}\\d{1,3} 172\\3[0-1]{1}\\d{1,3}\\d{1,3} 0:0:0:0:0:0:1 ::1
server.aas.protocol-header	包含傳入協議的首部，通常稱為“X-Forwarded-Proto”	
server.aas.protocol-header-https-value	協議頭的值，表示傳入的請求是否使用SSL	https
server.aas.port-header	用於覆蓋原始端口值的HTTP頭的名稱	X-Forwarded-Port
server.aas.remote-ip-header	從HTTP頭中提取遠端IP的名稱。例如，'X-FORWARDED-FOR'	
server.aas.basedir	應用伺服器基礎路徑，若不設置則使用臨時目錄	
server.aas.background-processor-delay	調用backgroundProcess方法之間的延遲	
server.aas.max-threads	最大線程數	200
server.aas.min-spare-threads	最小空閒線程數	25
server.aas.max-http-post-size	最大 Post 資料大小	104857600
server.aas.max-http-header-size	請求或響應的最大 http header 大小	8192
server.aas.max-swallow-size	請求體大小限制	20971520
server.aas.redirect-context-root	設置通過在路徑後面新增一個/來重定向到上下文根目錄的請求	
server.aas.use-relative-redirects	調用sendRedirect生成的HTTP 1.1及以後的位置首部是使用相對重定向還是絕對重定向， true/false	
server.aas.uri-encoding	URI 的編碼方式	utf-8
server.aas.max-connections	最大併發個數	10000
server.aas.accept-count	已連線但應用伺服器未處理數	100
server.aas.max-keep-alive-timeout	連線保活超時時間	30000ms
server.aas.max-keep-alive-requests	最多處理多少個請求後關閉連線， -1 表示不限制	100
server.aas.compression	是否啟用響應壓縮	FALSE
server.aas.enable-lookups	是否啟用 DNS 查詢以將遠端 IP 地址解析為域名，在訪問日誌中記錄客戶端的主機名	false
server.aas.compressible-mime-type	需要壓縮的MIME型別列表，用逗號分隔	text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,application/json,application/xml
server.aas.disable-upload-timeout	是否停用上傳超時	false
server.aas.additional-tld-skip-patterns	啟動掃描 JSP 標籤庫（TLD）時，跳過指定的 jar 包；多個以逗號分隔	
server.aas.relaxed-path-chars	允許請求行中的特殊字元 其餘配置方式： 1. 系統環境變數：AAS_RELAXED_PATH_CHARS； 2. JVM引數：-Drelaxed-path-char 優先順序：系統環境變數 > JVM引數 > Spring boot配置文件，配置示例請閱讀附錄J	
server.aas.relaxed-query-chars	允許請求行中引數的特殊字元 其餘配置方式： 1. 系統環境變數：AAS_RELAXED_QUERY_CHARS； 2. JVM引數：-Drelaxed-query-char 優先順序：系統環境變數 > JVM引數 > Spring boot配置文件，配置示例請閱讀附錄J	
server.aas.use-send-file	是否使用 sendfile 功能	true
server.aas.scan-manifest	是否掃描mf文件	true
server.aas.scan-all-directories	是否掃描所有目錄	true

server.aas.scan-all-files	是否掃描所有文件	false
server.aas.scan-bootstrap-class-path	是否掃描Bootstrap類路徑	false
server.aas.scan-class-path	是否掃描類路徑	true
server.aas.head-case-sensitive	head是否大小寫敏感	true
server.aas.license-path	License 路徑	
server.forward-headers-strategy	處理X-Forwarded-* headers的策略	
server.shutdown	伺服器支援的關閉型別	immediate
server.http2.enabled	是否啟用http2	false
server.aas.processor-cache	可以保留在快取中並在後續請求中重用的空間處理器的最大數目,springboot2.4及以上版本支援	20000
server.aas.connection-timeout	連線超時時間,springboot2.4及以上版本支援	
server.aas.remoteip.internal-proxies	匹配可信代理的正規表示式,springboot2.4及以上版本支援	10\.\d{1,3}\.\d{1,3}\.\d{1,3} 192\.168\.\d{1,3}\.\d{1,3} 169\.254\.\d{1,3}\.\d{1,3} 127\.\d{1,3}\.\d{1,3}\.\d{1,3} 172\.[16-9]{1}\.\d{1,3}\.\d{1,3} 172\.[2[0-9]{1}\.\d{1,3}\.\d{1,3} 172\.[3[0-1]{1}\.\d{1,3}\.\d{1,3} 0:0:0:0:0:0:1 ::1
server.aas.remoteip.protocol-header	包含傳入協議的首部，通常稱為“X-Forwarded-Proto”,springboot 2.4及以上版本支援	
server.aas.remoteip.protocol-header-https-value	協議頭的值，表示傳入的請求是否使用SSL,springboot2.4及以上版本支援	https
server.aas.remoteip.host-header	遠端主機의HTTP標頭的名稱,springboot2.4及以上版本支援	X-Forwarded-Host
server.aas.remoteip.port-header	用於覆蓋原始端口值的HTTP頭的名稱,springboot2.4及以上版本支援	X-Forwarded-Port
server.aas.remoteip.remote-ip-header	從HTTP頭中提取遠端IP的名稱。例如，`X-FORWARDED-FOR`。springboot2.4及以上版本支援	
server.aas.mbeanregistry.enabled	是否啟用AAS的MBean註冊中心,springboot2.4及以上版本支援	false
server.aas.thread.max	最大線程數,springboot2.4及以上版本支援	200
server.aas.thread.min-spare	最小線程數,springboot2.4及以上版本支援	25

12.9 附錄I 嵌入式部署日誌引數

引數	說明	默認值
server.aas.accesslog.enabled	是否列印 accesslog 日誌，為 true 時為列印	false
server.aas.accesslog.pattern	日誌格式	common
server.aas.accesslog.directory	日誌文件存放的路徑	無
server.aas.accesslog.prefix	日誌名稱字首	access_log
server.aas.accesslog.suffix	日誌名稱字尾	.log
server.aas.accesslog.rotate	是否啟用日誌輪轉	true
server.aas.accesslog.rename-on-rotate	是否推遲在檔名中包含日期戳，直到輪換	false
server.aas.accesslog.request-attributes-enabled	是否記錄請求的IP地址、主機名、協議和端口	false
server.aas.accesslog.file-date-format	日誌檔名中的日期格式	.yyyy-MM-dd
server.aas.accesslog.buffered	是否緩衝日誌記錄。設置 true 時，將非同步記錄日誌	true

server.aas.accesslog.condition-if	是否只有在“ServletRequest.getAttribute (conditionIf) ”不返回null時才啟用請求日誌記錄。springboot2.4及以上版本支援	
server.aas.accesslog.condition-unless	是否只有在“ServletRequest.getAttribute (conditionUnless) ”返回null時才啟用請求日誌記錄。springboot2.4及以上版本支援	
server.aas.accesslog.encoding	日誌文件使用的字元集,springboot2.4及以上版本支援	
server.aas.accesslog.locale	用於格式化日誌條目和日誌檔名字尾中的時間戳的區域設置,springboot2.4及以上版本支援	
server.aas.accesslog.check-exist	是否檢查日誌文件是否存在，以便在外部程序重新命名它時可以重新建立它,springboot2.4及以上版本支援	false
server.aas.accesslog.ipv6-canonical	是否使用RFC 5952定義的IPv6規範表示格式,springboot2.4及以上版本支援	false
server.aas.accesslog.max-days	刪除訪問日誌文件前保留的天數,springboot2.4及以上版本支援	-1

12.10 附錄J relaxed-path-chars和relaxed-query-chars配置示例

12.10.1 Linux

12.10.1.1 環境變數

```
export AAS_RELAXED_PATH_CHARS="\\"<>[\\]\^`{ }"
export AAS_RELAXED_QUERY_CHARS="\\"<>[\\]\^`{ }"
```

12.10.1.2 JVM引數

```
JAVA_OPTS="$JAVA_OPTS $JSSE_OPTS -Drelaxed-path-chars='\\"<>[\\]\^`{ }' -Drelaxed-query-chars='\\"<>[\\]\^`{ }' "
```

12.10.2 Spring boot配置文件

12.10.2.1 application.properties

```
server.aas.relaxed-path-chars=\\,<,>,[,],\\,^,`,{,},) # 允許特殊字元
server.aas.relaxed-query-chars=\\,<,>,[,],\\,^,`,{,},) # 允許特殊字元
```

12.10.2.2 application.yaml

```
server:
  aas:
    relaxed-path-chars: "\\"<>[\\],^`{ } # 允許特殊字元
    relaxed-query-chars: "\\"<>[\\],^`{ } # 允許特殊字元
```

12.11 附錄K actuator詳細說明

前置說明：嵌入式 starter 不自帶 actuator 模組，使用本附錄功能前需先在 pom 中自行引入 `spring-boot-starter-actuator` 依賴（參考「容器功能說明-監控支援」章節）。

12.11.1 配置

引數	說明	默認值
management.endpoints.web.exposure.include	選擇暴露端口(*為暴露所有端口); 輸入端口1,端口2,...,端口名n即可選擇性暴露多個端口	Health,info
management.endpoints.web.exposure.exclude	隱藏端口; 輸入端口1,端口2,...,端口名n即可選擇性隱藏多個端口	

management.server.port	actuator開放的端口，它用於提供應用程式的健康狀況、性能指標等管理資訊。通常情況下用於springboot2.X	8080
management.port	actuator開放的端口，它用於提供應用程式的健康狀況、性能指標等管理資訊。通常情況下用於springboot1.X	8080
management.context-path	通過更改此屬性的值，您可以更改管理訪問actuator的 URL 路徑。例如，如果您將 management.context-path 設置為 /management，則應用程式的管理端點 URL 將變為 http://localhost:8080/management/health，用於Spring boot 1.x	/actuator
management.endpoints.web.base-path	與management.context-path作用相同，用於Spring boot 2.x	/actuator
management.security.enabled	如果將其設置為 true，則需要進行身份驗證才能訪問管理端點。如果將其設置為 false，則可以無需身份驗證即可訪問管理端點。默認情況下，此屬性的值為 false。啟用此屬性可以幫助保護管理端點免受未經授權的訪問	false

12.11.2 端口

路徑	說明	詳情
/actuator	actuator服務訪問上下文	
/actuator/beans	beans 端點提供有關應用程式 bean 的資訊	詳見beans
/actuator/caches	檢視應用程式的快取	詳見caches
/actuator/health	檢視應用程式的健康狀態	詳見health
/actuator/info	提供有關應用程式的一般資訊（可自定義）	
/actuator/conditions	提供有關配置和自動配置類條件評估的資訊	詳見conditions
/actuator/configprops	提供有關應用程式@ConfigurationPropertiesBean 的資訊	詳見configprops
/actuator/env	提供有關應用程式的整個環境	詳見env
/actuator/loggers	提供對應用程式記錄器及其級別配置的訪問	詳見loggers
/actuator/heapdump	提供來自應用程式 JVM 的heap dump	響應是HPROF格式的二進位制資料
/actuator/threaddump	提供來自應用程式 JVM 的thread dump	詳見threaddump
/actuator/metrics	提供對應用程式指標的訪問	詳見metrics
/actuator/scheduledtasks	提供有關應用程式計劃任務的資訊	詳見scheduledtasks
/actuator/mappings	提供有關應用程式請求對映的資訊	詳見mappings

12.11.2.1 beans

自定義一個bean，然後執行程式後，進入 /actuator/beans 搜尋test即可檢視到test的相關資訊。

```
@Configuration
public class HelloWorld {
    @Bean
    public void test() {
        System.out.println("hello, world!");
    }
}
```

```
{
  "context":{
    "application":{
      "beans":{
        ....
      }
    }
  }
```

```

    "test": {
        "aliases": [],
        "scope": "singleton",
        "type": "void",
        "resource": "class path resource [com/apusic/demo/bean/HelloWorld.class]",
        "dependencies": []
    },
    ....
}
}
}
}

```



詳細引數介紹如下:

路徑	型別	說明
contexts	Object	由 id 鍵控的應用程式上下文
contexts.*.parentId	String	父應用程式上下文的 ID (如果有)
contexts.*.beans	Object	應用程式上下文中的 Bean 按名稱鍵入
contexts.*.beans.*.aliases	Array	任何別名的名稱
contexts.*.beans.*.scope	String	Bean 的範圍
contexts.*.beans.*.type	String	完全合格的 bean 型別
contexts.*.beans.*.resource	String	定義 bean 的資源 (如果有)
contexts.*.beans.*.dependencies	Array	任何依賴項的名稱

12.11.2.2 health

修改配置資訊顯示health端口的詳細內容。

```
management.endpoint.health.show-details=always # 默認為never
```

此時再次訪問/actuator/health端口檢視詳細內容，只有當components中所有元件均為UP，應用程式的健康狀態才為UP。（eg. 當新增數據源但是資料庫服務未開啟時元件中有關服務會顯示DOWN，此時應用程式的健康狀態為DOWN）

```

{
    "status": "UP",
    "components": {
        "diskSpace": {

```

```

        "status": "UP",
        "details": {
            "total": 107371032576,
            "free": 97080827904,
            "threshold": 10485760,
            "exists": true
        }
    },
    "ping": {
        "status": "UP"
    }
}
}

```

12.11.2.3 conditions

進入 `/actuator/conditions` 即可看到類似以下資訊的響應

```

{
  "contexts" : {
    "application" : {
      "positiveMatches" : {
        ...
        "EndpointAutoConfiguration#endpointOperationParameterMapper" : [ {
          "condition" : ...,
          "message" : ...
        } ],
        "GsonHttpMessageConvertersConfiguration.GsonHttpMessageConverterConfiguration" : {
          "notMatched" : [ {
            "condition" : ...,
            "message" : ...
          } ],
          "matched" : [ ]
        },
        ...
      },
      "unconditionalClasses" : [
        ...
      ]
    }
  }
}

```

響應結構如下:

路徑	型別	描述
<code>contexts</code>	Object	由 id 鍵控的應用程式上下文。
<code>contexts.*.positiveMatches</code>	Object	具有匹配條件的類和方法。
<code>contexts.*.positiveMatches.*.[]condition</code>	String	條件名稱。
<code>contexts.*.positiveMatches.*.[]message</code>	String	條件匹配的原因的詳細資訊。
<code>contexts.*.negativeMatches</code>	Object	條件不匹配的類和方法。
<code>contexts.*.negativeMatches.*.notMatched</code>	Array	符合的條件。
<code>contexts.*.negativeMatches.*.notMatched.[]condition</code>	String	條件名稱。

contexts.*.negativeMatches.*.notMatched.[] .message	String	條件不匹配的詳細資訊。
contexts.*.negativeMatches.*.matched	Array	符合的條件。
contexts.*.negativeMatches.*.matched.[] .condition	String	條件名稱。
contexts.*.negativeMatches.*.matched.[] .message	String	條件匹配的原因的詳細資訊。
contexts.*.unconditionalClasses	Array	無條件自動配置類的名稱（如果有）。
contexts.*.parentId	String	父應用程式上下文的 ID（如果有）。

12.11.2.4 configprops

進入 /actuator/configprops 即可看到類似以下資訊的響應。

```
{
  "contexts": {
    "application": {
      "beans": {
        "management.endpoints.web-org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties": {
          "prefix": "management.endpoints.web",
          "properties": {
            "pathMapping": {},
            "exposure": {
              "include": [
                "*"
              ],
              "exclude": []
            },
            "basePath": "/actuator"
          },
          "inputs": {
            "pathMapping": {},
            "exposure": {
              "include": [
                {
                  "value": "*",
                  "origin": "class path resource [application.properties] - 1:43"
                }
              ],
              "exclude": []
            },
            "basePath": {}
          }
        }
      }
    }
  }
}
```

結合我們的配置文件，從紅框中可以看到我們在配置文件中對應用程式的相關配置。

```
management.endpoints.web.exposure.include=*
```

響應詳情如下：

路徑	型別	描述
contexts	Object	由 id 鍵控的應用程式上下文。
contexts.*.beans.*	Object	@ConfigurationProperties 通過 bean 名稱鍵入的 beans。
contexts.*.beans.*.prefix	String	應用於 bean 屬性名稱的字首。
contexts.*.beans.*.properties	Object	bean 的屬性作為名稱-值對。
contexts.*.beans.*.inputs	Object	繫結到此 bean 時使用的配置屬性的來源和值。
contexts.*.parentId	String	父應用程式上下文的 ID（如果有）。

12.11.2.5 env

進入 /actuator/env 路徑下，即可看到如下頁面：

```
{
  "activeProfiles": [],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": "Complex property type java.lang.Integer"
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "java.runtime.name": {
          "value": "OpenJDK Runtime Environment"
        },
        "spring.output.ansi.enabled": {
          "value": "always"
        },
        "sun.boot.library.path": {
          "value": "D:\\workspace\\jdk8\\jre\\bin"
        },
        "java.vm.version": {
          "value": "25.382-b05"
        }
      }
    }
  ]
}
```

從圖中可以看到我們一些系統配置以及服務端口值的型別

路徑	型別	描述
property	Object	來自環境的屬性（如果發現）
property.source	String	屬性來源的名稱
property.value	String	屬性的價值
activeProfiles	Array	活動配置文件的名稱（如果有）
propertySources	Array	屬性來源按優先順序排列
propertySources.[].name	String	屬性來源的名稱
propertySources.[].property	Object	屬性來源中的屬性
propertySources.[].property.value	Varies	屬性價值
propertySources.[].property.origin	String	屬性來源

12.11.2.6 loggers

進入 `actuator/loggers` 路徑可以看到以下資訊。



```
"levels": [
  "OFF",
  "ERROR",
  "WARN",
  "INFO",
  "DEBUG",
  "TRACE"
],
"loggers": {
  "ROOT": {
    "configuredLevel": "INFO",
    "effectiveLevel": "INFO"
  },
  "com": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  },
  "com.apusic": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  },
  "com.apusic.ams": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  }
},
```

從圖中可以看到日誌的相關等級，不同名稱的記錄器的有效級別，響應格式如下。

路徑	型別	描述
levels	Array	日誌系統支援的級別
loggers	Object	按名稱鍵入的記錄器
groups	Object	按名稱鍵入的記錄器組
loggers.*.configuredLevel	String	記錄器的配置級別（如果有）
loggers.*.effectiveLevel	String	記錄器的有效級別
groups.*.configuredLevel	String	記錄器組的配置級別（如果有）
groups.*.members	Array	屬於該組的記錄器

可以通過 `/actuator/loggers/{logger.name}` 訪問單個記錄器的相關資訊

該端口下還有以下介面

路徑	描述	
<code>/actuator/loggers/{logger.name}</code>	訪問單個記錄器的相關資訊	
<code>/actuator/loggers/{group.name}</code>	訪問某個組下的記錄器的相關資訊	

12.11.2.6.1 設置日誌級別

設置單個記錄器

示例：選擇 `com.apusic` 這個記錄器進行設置，在設置前進入<http://localhost:9999/actuator/loggers/com.apusic>可以檢視當前記錄器的相關資訊。

HTTP <http://localhost:9999/actuator/loggers/com.apusic>

GET <http://localhost:9999/actuator/loggers/com.apusic>

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

	Key
	Key

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "configuredLevel": null,
3    "effectiveLevel": "INFO"
4  }

```

然後修改請求為 POST，在Headers和Body中分別加入Content-Type: application/json和{"configuredLevel":"debug"}，當Body中為空時即為清除日誌級別。

HTTP <http://localhost:9999/actuator/loggers/com.apusic>

POST <http://localhost:9999/actuator/loggers/com.apusic>

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Headers 8 hidden

	Key	Value
☒	Content-Type	application/json
	Key	Value

HTTP <http://localhost:9999/actuator/loggers/com.apusic>

POST <http://localhost:9999/actuator/loggers/com.apusic>

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

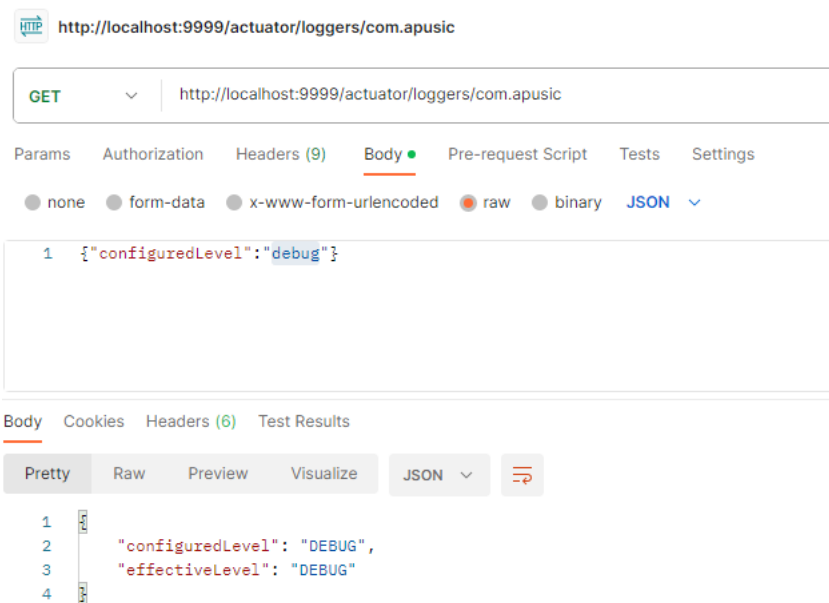
● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON

```

1  {"configuredLevel":"debug"}

```

然後傳送請求後，再次傳送GET請求可以看到日誌級別已經變更為DEBUG。



HTTP http://localhost:9999/actuator/loggers/com.apusic

GET http://localhost:9999/actuator/loggers/com.apusic

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1 {\"configuredLevel\": \"debug\"}
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

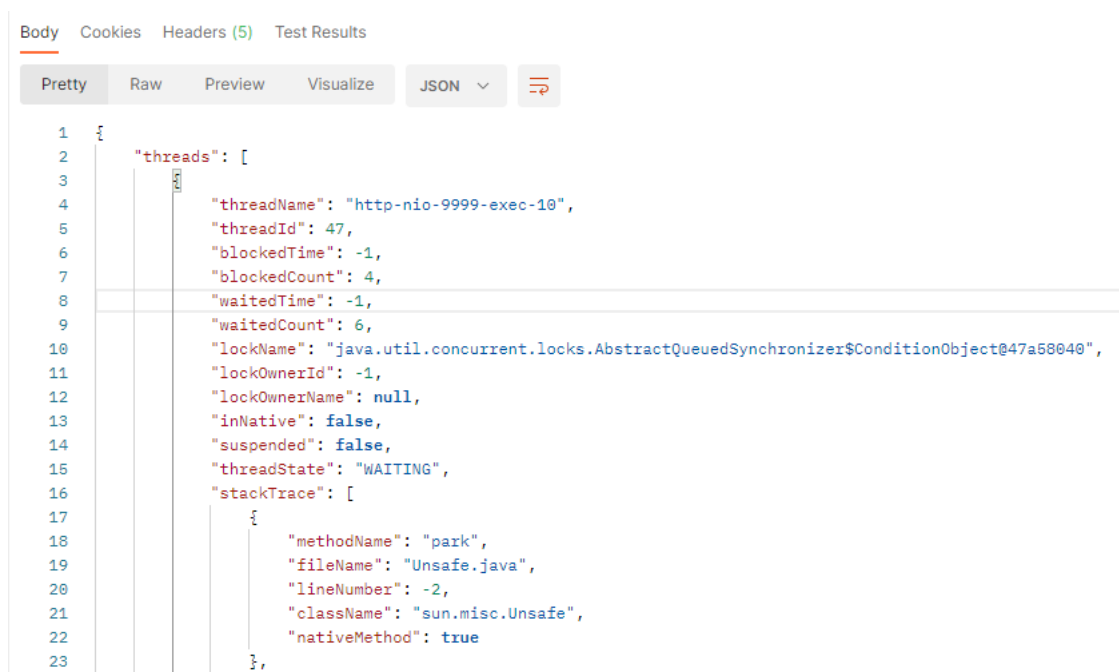
```
1 {
2   \"configuredLevel\": \"DEBUG\",
3   \"effectiveLevel\": \"DEBUG\"
4 }
```

設置組的日誌級別

與設置單個記錄器同理，只需要將路徑中的 `{logger.name}` 替換為 `{group.name}` 即可。

12.11.2.7 threaddump

進入 `/actuator/threaddump` 之後可以看到以下頁面



Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   \"threads\": [
3     {
4       \"threadName\": \"http-nio-9999-exec-10\",
5       \"threadId\": 47,
6       \"blockedTime\": -1,
7       \"blockedCount\": 4,
8       \"waitedTime\": -1,
9       \"waitedCount\": 6,
10      \"lockName\": \"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject@47a58040\",
11      \"lockOwnerId\": -1,
12      \"lockOwnerName\": null,
13      \"inNative\": false,
14      \"suspended\": false,
15      \"threadState\": \"WAITING\",
16      \"stackTrace\": [
17        {
18          \"methodName\": \"park\",
19          \"fileName\": \"Unsafe.java\",
20          \"lineNumber\": -2,
21          \"className\": \"sun.misc.Unsafe\",
22          \"nativeMethod\": true
23        }
24      ]
25    }
26  ]
27 }
```

12.11.2.8 metrics

進入 `/actuator/metrics` 之後可以看到以下頁面

```
{
  \"names\": [
    \"http.server.requests\",
    \"jvm.buffer.count\",
    \"jvm.buffer.memory.used\",
    \"jvm.buffer.total.capacity\",
    \"jvm.classes.loaded\",
```

```

    "jvm.classes.unloaded",
    "jvm.gc.live.data.size",
    "jvm.gc.max.data.size",
    "jvm.gc.memory.allocated",
    "jvm.gc.memory.promoted",
    "jvm.gc.pause",
    "jvm.memory.committed",
    "jvm.memory.max",
    "jvm.memory.used",
    "jvm.threads.daemon",
    "jvm.threads.live",
    "jvm.threads.peak",
    "jvm.threads.states",
    "logback.events",
    "process.cpu.usage",
    "process.start.time",
    "process.uptime",
    "system.cpu.count",
    "system.cpu.usage"
  ]
}

```

以 `/actuator/metrics/{metric.name}` 方式傳送GET請求，可以得到具體的指標的資訊，以 `jvm.gc.max.data.size` 為例，瀏覽器會輸出以下響應資訊。

```

{
  "name": "jvm.gc.max.data.size",
  "description": "Max size of long-lived heap memory pool",
  "baseUnit": "bytes",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 1.431830528E9
    }
  ],
  "availableTags": []
}

```

響應結構如下:

路徑	型別	描述
name	String	指標名稱
description	String	指標描述
baseUnit	String	公制的基本單位
measurements	Array	指標的測量
measurements[].statistic	String	測量統計。（TOTAL、、、、、、、）。TOTAL_TIME_COUNT___MAX``VALUE``UNKNOWN``ACTIVE_TASKS``DURATION
measurements[].value	Number	測量值
availableTags	Array	可用於向下鑽取的標籤
availableTags[].tag	String	標籤的名稱
availableTags[].values	Array	標籤的可能值

12.11.2.8.1 所有指標相關說明

指標	說明
http.server.requests	提供了有關應用程式處理的每個HTTP請求的資訊，如請求計數、請求持續時間等
jvm.buffer.count	提供了有關JVM緩衝區使用情況的資訊，如緩衝區數量、緩衝區使用率等
jvm.buffer.memory.used	JVM 堆外緩衝區（Direct/Mapped）已用記憶體，單位 bytes
jvm.buffer.total.capacity	緩衝區總容量
jvm.classes.loaded	已載入類數
jvm.classes.unloaded	已解除安裝類總數
jvm.gc.live.data.size	活躍對象的大小，可以幫助開發人員識別記憶體洩漏問題和調整JVM堆大小
jvm.gc.max.data.size	堆的最大大小，可以幫助開發人員瞭解JVM堆的大小限制，並根據需要進行調整
jvm.gc.memory.allocated	已分配記憶體的總量，這可以幫助開發人員瞭解應用程式的記憶體使用情況，並進行優化
jvm.gc.memory.promoted	通過垃圾回收已提升的對象大小，可以幫助開發人員瞭解垃圾回收器的性能，並調整堆大小和垃圾回收策略
jvm.gc.pause	垃圾回收暫停的次數和持續時間，可以幫助開發人員瞭解垃圾回收器的性能，並調整堆大小和垃圾回收策略
jvm.memory.committed	已提交的記憶體量，可以幫助開發人員瞭解應用程式的記憶體使用情況，並進行優化
jvm.memory.max	最大可用記憶體量，可以幫助開發人員瞭解JVM堆的大小限制，並根據需要進行調整
jvm.memory.used	已使用的記憶體量，可以幫助開發人員瞭解應用程式的記憶體使用情況，並進行優化
jvm.threads.daemon	守護線程的數量，可以幫助開發人員瞭解應用程式中守護線程的使用情況，並進行優化
jvm.threads.live	活動線程的數量，以幫助開發人員瞭解應用程式中線程的使用情況，並進行優化
jvm.threads.peak	峰值線程數，可以幫助開發人員瞭解應用程式中線程使用的最高峰值，並進行優化
jvm.threads.states	每種線程狀態的數量，可以幫助開發人員瞭解應用程式中線程狀態的分佈情況，並進行優化
logback.events	Logback日誌事件的數量，可以幫助開發人員瞭解應用程式中日誌事件的數量和型別，並進行優化
process.cpu.usage	程序CPU使用率，可以幫助開發人員瞭解應用程式在CPU方面的性能，並進行優化
process.start.time	程序啟動時間，可以幫助開發人員瞭解應用程式執行時間和穩定性
process.uptime	程序執行時間，可以幫助開發人員瞭解應用程式執行時間和穩定性
system.cpu.count	系統CPU核心數，可以幫助開發人員瞭解系統硬體規格，並優化應用程式性能
system.cpu.usage	系統CPU使用率，可以幫助開發人員瞭解系統在CPU方面的性能，並進行優化

12.11.2.9 scheduledtasks

給 `/actuator/scheduledtasks` 傳送GET請求，可以看到如下介面

Body	Cookies	Headers (5)	Test Results
Pretty	Raw	Preview	Visualize
JSON			
1			
2			"cron": [],
3			"fixedDelay": [],
4			"fixedRate": [],
5			"custom": []
6			

此時我們沒有設置定時任務，響應的結構如下：

路徑	型別	描述
cron	Array	Cron任務（如果有）。
cron[].runnable.target	String	將要執行的目標。
cron[].expression	String	Cron表示式。

fixedDelay	Array	修復延遲任務（如果有）。
fixedDelay.[].runnable.target	String	將要執行的目標。
fixedDelay.[].initialDelay	Number	第一次執行之前的延遲（以毫秒為單位）。
fixedDelay.[].interval	Number	上次執行結束與下一次執行開始之間的時間間隔（以毫秒為單位）。
fixedRate	Array	固定時間任務（如果有）。
fixedRate.[].runnable.target	String	將要執行的目標。
fixedRate.[].interval	Number	每次執行開始之間的時間間隔（以毫秒為單位）。
fixedRate.[].initialDelay	Number	第一次執行之前的延遲（以毫秒為單位）。
custom	Array	具有自定義觸發器的任務（如果有）。
custom.[].runnable.target	String	將要執行的目標。
custom.[].trigger	String	觸發任務。

12.11.2.10 mappings

給 `/actuator/mappings` 傳送GET請求，可以看到如下介面

```

{
  "contexts": {
    "application": {
      "mappings": {
        "dispatcherServlets": {
          "dispatcherServlet": [
            {
              "handler": "Actuator web endpoint 'loggers-name'",
              "predicate": "{POST [/actuator/loggers/{name}], consumes [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className": "org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljava/lang/HttpServletRequest;Ljava/util/Map;Ljava/lang/Object;)"
                },
                "requestMappingConditions": {
                  "consumes": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            }
          ]
        }
      }
    }
  }
}
```

全國統一服務熱線
4008-555-800

金蝶天燕雲計算股份有限公司（簡稱“金蝶天燕”）成立於2000年，前身為“金蝶中間件公司”，是金蝶集團旗下基礎軟件平台提供商，雲計算國家標準制定企業，國家信創產業核心軟件企業。金蝶天燕是國家863重點研發計劃與核高基重大專項承接企業，也是“兩網一站四庫十二金”國家重點工程的基礎平台提供商，產品廣泛應用於政府、軍工、金融、能源等關鍵行業，累計服務客戶總數超過10萬家。



雲計算國家標準制定企業
金蝶集團旗下基礎軟件企業
資訊技術應用創新核心企業
官網：www.apusic.com

